TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

EETU PREHTI
DEVELOPMENT OF A LABORATORY ENTRY SYSTEM

Master of Science Thesis

Examiner: Prof. Hannu Koivisto
Examiner and topic approved by the
Faculty Council of the Faculty of
Engineering sciences
on 26 April 2017

ABSTRACT

EETU PREHTI: Development of a laboratory entry system
Tampere University of Technology
Master of Science Thesis, 67 pages, 10 Appendix pages
April, 2018
Master's Degree Programme in Automation Technology
Major: Information Systems in Automation
Examiner: Prof. Hannu Koivisto
Keywords: Valmet DNA, pulp, laboratory, web development, statistical process control, virtualization

Quality management is one of the hardest tasks on any manufacturing process. Often quality control focuses on checking that the end product complies with its specifications, even though the real problems are somewhere in the manufacturing process. Thus, the most efficient method for successful quality management is to monitor, control and improve the process itself.

Challenges in quality management have also been recognized in pulp industry, where the process is monitored continuously by taking measurements with automatic analyzers. Some process attributes, however, need to be monitored by a laboratory, which takes periodic samples, analyzes them and inserts results to database for further inspection. One of the most important monitoring methods where analyses can be utilized is *statistical process control* (SPC), where collected data is used with statistical methods for determining the process state.

The goal of this thesis work was to implement a new application for Valmet Automation with main focus on supporting laboratory work by providing features for inserting analysis values and monitoring the process with integrated SPC features. Research done during the thesis work included reviewing the theory of SPC, interviewing selected pulp mill laboratory staff members and Valmet employees, and finally exploring suitable development techniques for the application.

The results chapter consists of the general overview and validation of the application and evaluation of the development project as a whole. Finally, some of the most important future development tasks were planned to be done after the thesis project was concluded.

TIIVISTELMÄ

EETU PREHTI: Laboratorioanalyysien syöttöjärjestelmän kehitys
Tampereen teknillinen yliopisto
Diplomityö, 67 sivua, 10 liitesivua
Huhtikuu 2018
Automaatiotekniikan koulutusohjelma
Pääaine: Automaation tietotekniikka
Tarkastajat: Prof. Hannu Koivisto
Avainsanat: Valmet DNA, sellu, laboratorio, web-sovellus, tilastollinen prosessinohjaus, virtualisointi

Laadunhallinta on monen valmistusprossin yksi haastavimmista osa-alueista. Usein laadunhallinnan menetelmät keskittyvät tarkastelemaan pelkästään lopputuotteen asettumista määrittelyn sallimiin toleransseihin, vaikka todellinen laatuongelmien aiheuttaja on itse valmistusprosessissa. Tästä johtuen tehokkain tapa hallita laatua onkin valvoa ja parantaa itse valmistusprosessia.

Laadunhallintaan liittyvät kysymykset tunnetaan hyvin myös selluteollisuudessa, jossa valmistusprosessia valvotaan säännöllisesti näytteitä ottavilla analysaattoreilla. Joitain prosessiparametreja ei kuitenkaan voida valvoa näin, vaan tähän tehtävään tarvitaan laboratoriota, jonka tehtävänä on ottaa säännöllisesti näytteitä, syöttää ne tietokantaan jatkotarkastelua varten, sekä tarkkailla prosessia tilastollisen prosessinohjauksen (Statistical Process Control, SPC) menetelmiä hyödyntäen. Lyhyesti kuvattuna tilastollinen prosessinohjaus hyödyntää tilastomatematiikan metodeja prossin tilan seuraamiseksi.

Tämän diplomityön päätavoitteena oli toteuttaa Valmet Automationille sovellus analyysiarvojen syöttämiseksi, kommentoimiseksi, sekä prosessin tilan seuraamiseksi tilastollisen prosessinohjauksen menetelmiä hyödyntäen. Diplomityön aikana tehty tutkimustyö koostui SPC-teorian tarkastelusta, valittujen sellutehtaiden laboratoriohenkilökunnan, sekä Valmetin työntekijöiden haastatteluista ja sovelluskehitykseen soveltuvien tekniikoiden valitsemisesta.

Työn tulokset koostuvat valmiin sovelluksen esittelystä, sovelluksen validoinnista, sekä kehitysprojektin yleiskatsauksesta. Lisäksi esiteltiin tärkeimpiä diplomityön päättymisen jälkeen tehtäväksi jääviä kehitystöitä.

PREFACE

This thesis was written for Valmet Automation where I have also had the pleasure of working during my studies. Thesis writing took place between October 2015 and March 2018.

I wish to send my thanks to Heikki Turppo and Jaakko Oksanen for their advices and genuine interest towards the thesis writing process. I wish also to thank Hannu Koivisto for supervision and guidance during the work. I wish also to express my gratitude to all of my colleagues in Valmet Automation's Performance Execution Solutions team and Research and Development for their expertise and help. Special thanks go to my friends, family, and my girlfriend for supporting me during the many phases of writing the thesis.

Tampere, 07.03.2018

Eetu Prehti

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

AMD	Asynchronous Module Definition
API	Application Programming Interface
DLL	Dynamically Linkable Library
DNA	Dynamic Network of Applications
DI	Dependency Injection
DOM	Document Object Model
HMI	Human Machine Interface
IDE	Integrated Development Environment
IIS	Internet Information Services
JSON	JavaScript Object Notation
NPM	Node.js Package Manager
ODBC	Open Database Connectivity
OWASP	Open Web Application Security Project
R&D	Research and Development
RDF	Resource Description Framework
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
SE	Standard Error of Means
SPC	Statistical Process Control
SVG	Scalable Vector Graphics
XSS	Cross-Site Scripting

1. INTRODUCTION

Finnish economy has always relied on forests for producing pulp and paper products. Nowadays, pulp and paper industry has grown to great proportions. In the year 2016 the combined net value of Finnish pulp and paper exports' was roughly EUR 10 milliard [1]. However, last decade has required a lot of adapting from the industry, as the global trend of writing and news paper consumption has been declining, while paperboard and cardboard consumption has been on the rise. In order to match changing market demands, new type of pulp mills are being developed, which can produce a large selection of bio materials. One good example of the latest technology in pulp production is Metsä Group's new bio product mill at Äänekoski with invested value of EUR 1,2 milliard [2] [3].

Long history in pulp and paper production has brought Finland global recognition as a provider of top-quality pulp and paper. Quality, however, doesn't emerge on its own, but requires continuous effort from employees as well as collaboration between industry branches. First step in the path of successful quality management is thorough understanding of the manufacturing processes. Too often quality management focuses only on discarding defecting products by checking them against specifications. This approach is overly wasteful as pinpointing and fixing under performing sections of the manufacturing process would be much more effective approach. In other words, keeping the process stable over time and minimizing variations is the key to achieving consistent quality of the end product.

Instead of merely checking the end product, a better alternative for quality management is to directly monitor and control the manufacturing process by using a methodology of *statistical process control*, SPC. The main idea behind SPC is to provide a set of mathematical tools for determining process fitness instead of relying on human intuition alone. In practice, this happens by establishing so called control charts, where operators may follow process development over time and detect clear signals when intervention to normal operation is needed.

One essential requirement for implementing statistical process control is to collect periodically information about the process. Depending on the target process, this can be done with automatic analyzers, which collect data directly from the process. Some process attributes, however, can't be determined through automation. This is where laboratory has an essential role. By collecting and analyzing samples in the laboratory, a more complete understanding of the process can be achieved than using analyzers alone. Additionally, laboratory has another important role, which is monitoring the state of analyzers. This happens by taking samples from the same process as analyzer and comparing results for any differences.

Laboratory work can be a challenge itself. Organizing daily work, collaborating with team, keeping track of daily progress, inserting values, checking the process state and alerting for any irregularities can quickly become overwhelming. In order to ease some of these recurring tasks, several laboratory systems have been developed. One such application is *DNALab*, which has been developed by Valmet Automation and will have an essential role in this thesis.

DNALab is an application for managing analysis entries in an industrial laboratory and has been serving its users from the early 2000's. However, its current implementation is coming to the end of its life cycle. Development technologies, like Visual Basic, are no longer supported and have become increasingly laborious to maintain. Thus, the main purpose for this thesis work is to develop a new laboratory system, from here on referred by its work name as *Lab Entry*, for replacing DNALab. In order to succeed in this development work multiple aspects need to be taken into account. These are, for example, researching and implementing SPC features, planning application development process and collecting customer feedback and using it to form the application's functional requirements.

In the following chapters the thesis begins by reviewing one of the main reasons for taking laboratory analyses – statistical process control. Next chapter contains the functional and non-functional requirements for the application. Both have been formed based on interviews made with the customer and Valmet application specialists' interviews, and DNALab feedback collected throughout the years. Following chapter studies some of the most common aspects of web applications and their development. Next chapter contains research of technological options for achieving previously mentioned functional and non-functional requirements for the application. More precisely, this includes programming languages, design patters,

frameworks and application environments. Second to final chapter *Results* consists of general overview of Lab Entry application: what are its main features, how customer wishes were implemented and how application validation was conducted. Future development plans, which go beyond this thesis work, have also been addressed. Final chapter *Conclusions* sums whole thesis by highlighting the main research methods, key results, and findings during the work.

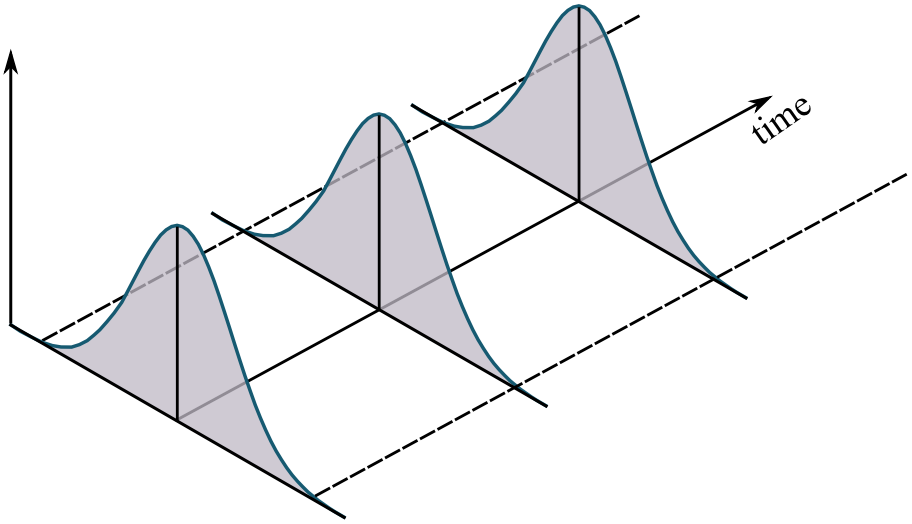
2. STATISTICAL PROCESS CONTROL

There's no doubt that the key factor for any company's success is satisfied customer. One of the requirements to customer satisfaction is quality, which can be defined as *fulfilling the needs of the customer*. Quality, however, doesn't come easily but requires hard work to maintain and improve. In many companies a lot of effort is directed to checking that the end product is in compliance with its specifications. Unfortunately, this approach is mostly wrong, since the defect has already happened somewhere along the manufacturing chain. Thus, identifying and managing efficiently any misbehaving processes is essential for efficient quality control. This is where statistical process control has an important role as it provides tools for improving and monitoring processes and gives guidelines when and how operator should intervene. More precisely, statistical process control measures variation, which is one of the most common reason for quality problems. [4, pp. 4, 16] This chapter contains an in-depth overview of statistical process control, which will have an essential role in Lab Entry application.

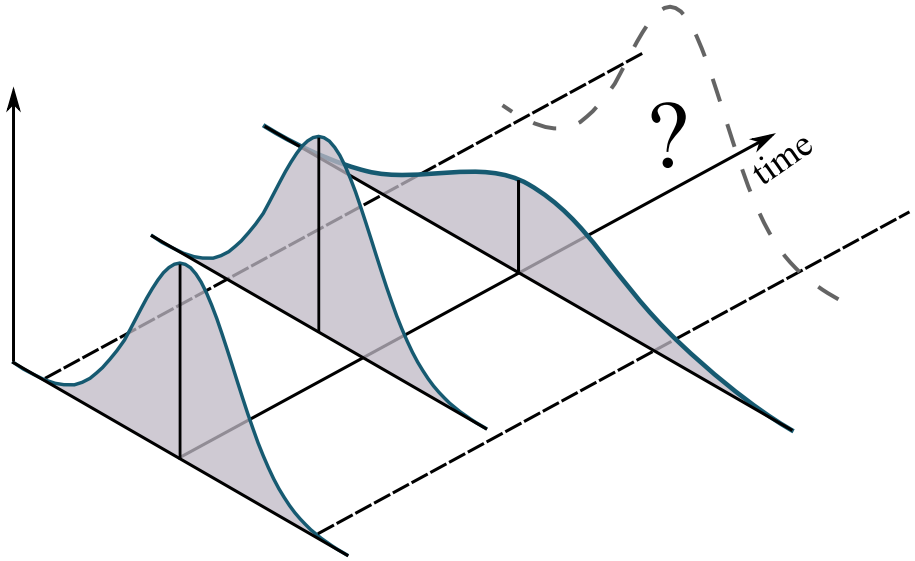
2.1 Variation sources and the four process states

Every process has always some level of variation affecting the end product, leading to the fact that two products are never exactly alike. This variation may be immeasurably small or large in magnitude but nevertheless always present. According to the father of statistical process control, Dr. Walter Stewhart, variation can be categorized as being either *controlled* or *uncontrolled*. Controlled variation remains stable and consistent over time, meaning that it is predictable and contributes to stable amount of non-conforming products. Controlled variation is often referred as *common cause* variation. Another type, uncontrolled variation, is unpredictable, and may lead to large amounts of non-conforming products at one moment while small amounts at another. Most importantly, uncontrolled variation can't be predicted. Uncontrolled variation is often called *special cause* or *assignable*

cause variation due to the fact that it is caused by an identifiable source, which is not originally part of the process. [5, p. 4] Both variation effects are shown in Figure 2.1.



Process A: Common cause variation stays stable over time.



Process B: Special cause variation causes unpredictability in the process.

Figure 2.1 *Effects of common (A) and special cause (B) variations on the process' distribution [5, p. 5].*

How much variation is acceptable is defined by the specification tolerances. Traditionally, the amount of non-conformance to specifications has been the only

measure of process fitness. In practice this means that process can have only two states, which are determined based on the process output: the first state is the normal state, where conformance is at 100% and the second state is when non-conformance can be detected. Actions are taken in such cases where non-conformance occurs, while otherwise the process is assumed to be performing fine. However, with the knowledge obtained by the statistical process control, this is not the complete truth. Introducing the state of control has profound impact on process fitness and conformance, leading to a total of four possible states for any process. [5, pp. 11-12] These states are shown in picture 2.2.

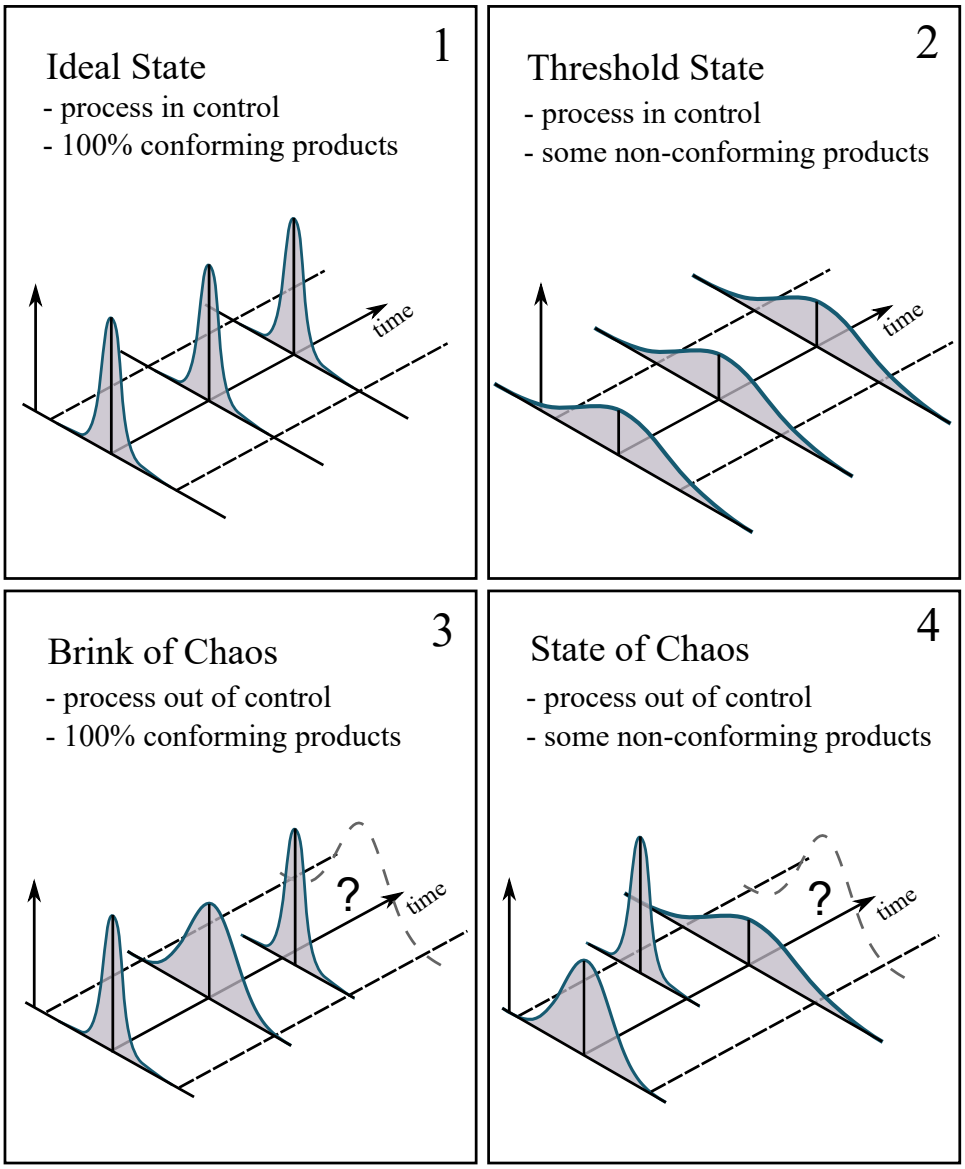


Figure 2.2 *The four states of a process [5, p. 15].*

As shown in the Figure 2.2, the best state for the process to be is in the *Ideal State*, where it has conformance of 100% and is in control. In this state no other actions need to be taken but only monitor for any problems which may lead to non-conformity or loss of control. The second *Threshold State* happens when some non-conformity occurs in the process while it still shows being in control. As the process is still in control, the amount of non-conformity remains stable over time. In order to bring the process back to the Ideal State, manufacturer needs to improve

the process by replacing worn parts or by investing in new machinery, hopefully reducing the spread. The third state, *Brink of Chaos*, means that the process has gone out of control due to an assignable cause. Still, even though being out of control, process is producing 100% conforming parts. However, this state doesn't usually last and the process tends to move to the *State of Chaos* due to the increasing effect of assignable cause(s). In the fourth state of Chaos, the process is producing non-conforming products with unpredictable quantities. The only way of bringing the process out of this state back to ideal state is by finding and removing the assignable cause. [5, pp. 12-16]

2.2 Accuracy and precision

Variation, which was briefly mentioned in the previous chapter, can affect process' distribution by two different ways: shifting of the process mean or increasing the magnitude of dispersion. In order to make it possible to have a clear understanding of what statistical process control is about, it is essential to make a distinction between these two. Shifting, or loss of accuracy, means that the process distribution is out of target value to either positive or negative direction. Spreading, or loss of precision, means that though average value of measurements is on the target, they have gotten more scattered around the target value. Both effects are shown in Figures 2.3 and 2.4. [4, pp. 73-75]

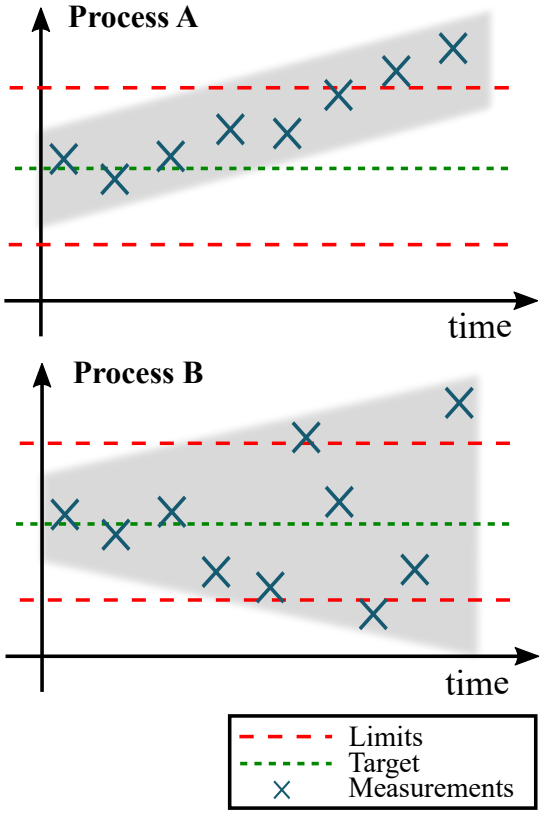


Figure 2.3 *Illustration of decrease in accuracy (Process A) and precision (Process B) over time [4, p. 324, 328].*

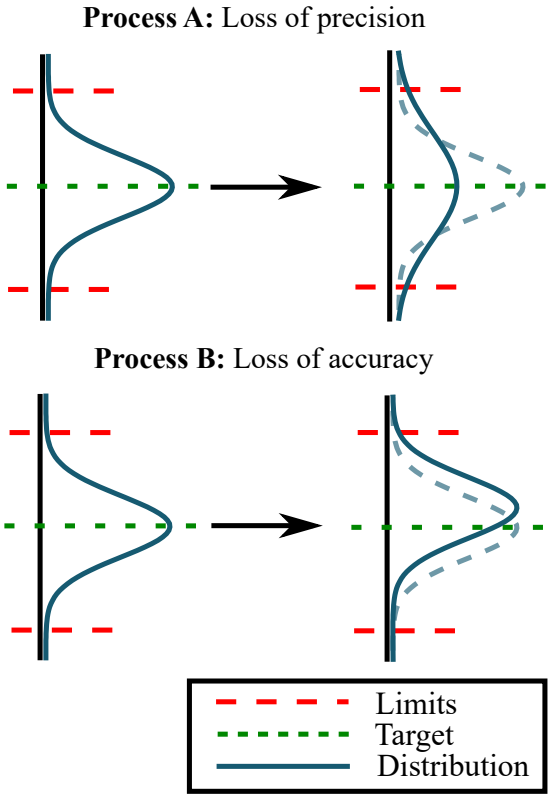


Figure 2.4 *Effects of different variations types on distribution over time. [4, pp. 78, 94, 96]*

As Figure 2.3 demonstrates, process A measurement values have started to climb towards the upper specification limit, eventually leading to a situation where values have moved out of specification tolerances. It is notable to mention that the dispersion (grey area) has remained the same. Process B demonstrates situation, where the degree of dispersion has increased (grey area), while process mean has remained the same. It is also possible for both effects to occur simultaneously depending on the process. Figure 2.4 shows the effects of spreading and shifting to process distribution. Although the example figure has normal distribution, this effect is also applicable to other type of distributions as well.

2.3 Data types, collection and subgrouping

The first step in any type of process control is to collect information from the process. Only this way it is possible to make rationalized adjustments on the process parameters. In the context of statistical process control data can be categorized in two ways: counting and measurement. An example of a counting, also known as attribute, data source is defect count n per 10 inspected samples, where defectiveness for a single sample is determined by two-way binary classification. The total count n is in its nature a whole number and provides discrete data. Second option for data collection is measurement and it provides variable data, where values can vary in a continuous scale. [4, p. 45]

In order to discover the actual situation within the process, the data collection should be planned carefully. This involves picking a suitable sample collection place and interval. For example, if the process is being ran in multiple operator shifts, samples need to be taken during each shift in order to discover variation between them. If process is divided in two or more parallel procedures, collecting samples from a common stock is not adequate enough to pinpoint faulty or under performing procedure. It should be kept in mind that data collection is always based on need, not ease of collection. [4, pp. 44-45]

Before statistical process control methods can be used, collected data has to be organized to so called subgroups. Subgroups are the basis of all statistical process control charts, as the key values *average*, *range* and *standard deviation*, are calculated based on them. Ideally, subgroups contain values from samples which represent the same condition within the process. Subgroup size is relatively free to choose, although recommendations for certain chart types exist. Main target for

subgrouping is to show the greatest similarity within each subgroup and the greatest difference among different subgroups. Same idea in terms of SPC is that subgroup size should be selected in such a way that it shows only common cause variation within the group, but detects special causes between the groups. [6] Examples of in-group variation and between-group variation are illustrated in Figure 2.5.

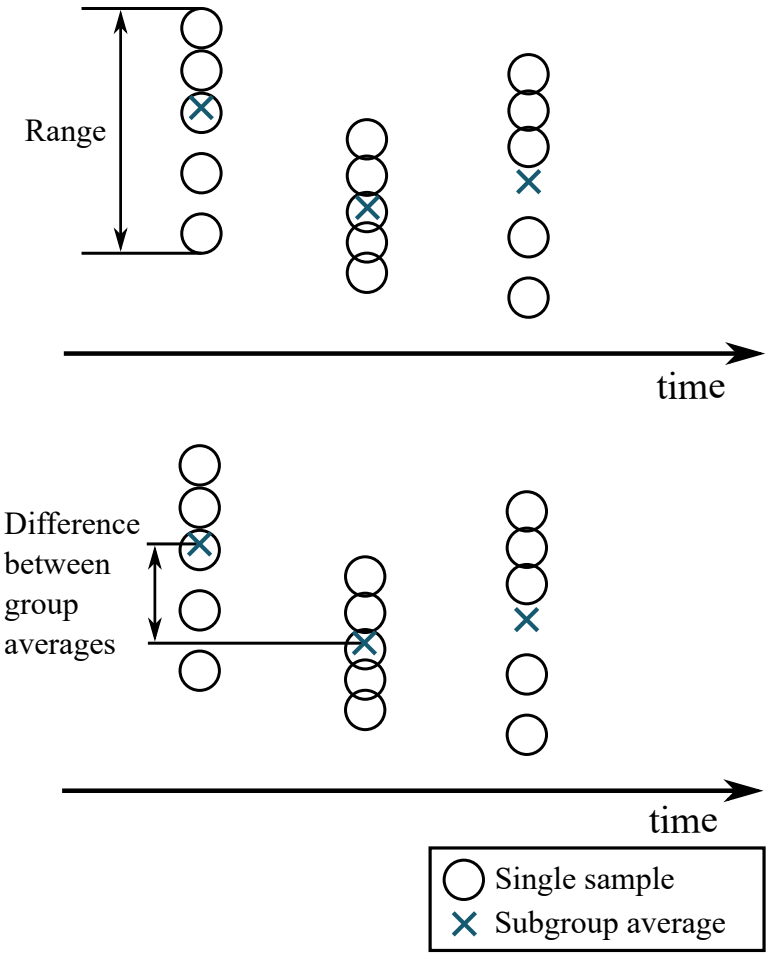


Figure 2.5 *Variation within group and between groups with group size of five. [6]*

Although group size is relatively free to choose, there might be some natural group size implied by the process. For example, if paint filling process is discharging to six cans simultaneously, it is natural to choose six, one from each nozzle, as the group size. This way it is possible to monitor each nozzle and the process as a whole simultaneously. [4, pp. 123-124]

2.4 Measures for variation

As stated previously, accuracy and precision are the most important indicators for monitoring process state. In order to have comparable values for these indicators, collected process measurements have to be refined through mathematical formulas. For accuracy there are three key attributes which tell us useful information about the process. These are *mean* (arithmetic average), *mode* and *median*. Mean can be calculated as follows: $mean = \sum_{n=1}^{\infty} x_i/n$, where x_i is single measurement and n is total count of measurements. Mode is determined as the most commonly occurring value within the sample set and median is the midmost value or average of two middle values for even number of values. If all of these parameters have the same value, distribution will be perfectly symmetric. In practice, however, distributions are more or less asymmetric (*skewed*), having different values for each median, mean, and mode. An approximate relation for all values can be calculated with $mean - mode = 3(mean - median)$. An example of a skewed distribution with key attributes is illustrated in Figure 2.6. [4, pp. 83-86]

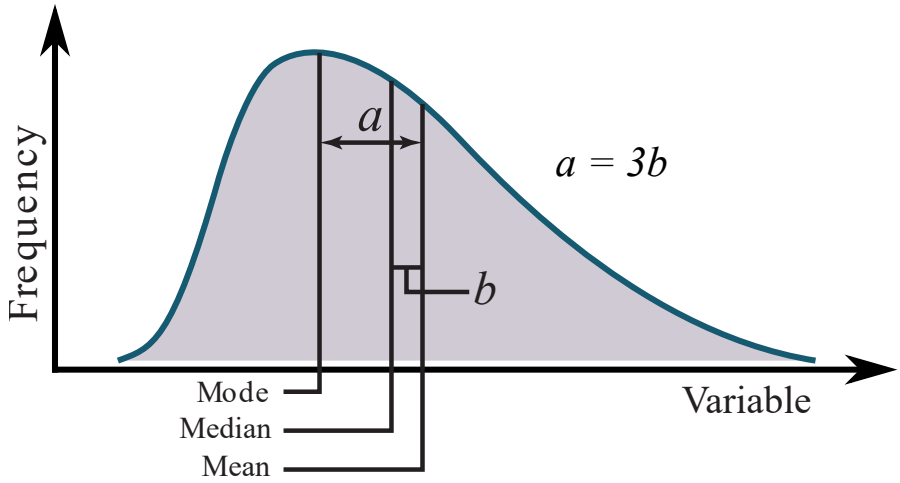


Figure 2.6 *Skewed distribution with relation between mean, median and mode [4, p. 86].*

Second key indicator, precision, can also be measured several different ways. The simplest method is to calculate range, which is the difference between the biggest and the smallest value. However, range has two major problems: its value tends to increase as the sample size increases and it doesn't give any information about the dispersion of the data points. Benefits for using range are its simplicity and ease of evaluation. Another way of representing the dispersion is standard deviation, which can be calculated with following formula:

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum(x_i - \overline{X})^2}{n}}, \tag{2.1}$$

where x_i is the value of sample i, n is the total number of samples, and $\overline{X}$ is the mean of all samples. However, it is noteworthy to mention that theoretical form of standard deviation is not suitable for real word situations as measurement count is always a subset of all measurements in the process history. Usually, sampled standard deviation tends to underestimate the standard deviation of the whole process. This effect is most notable with small samples. To correct for the bias, squared deviation sum is divided with $n - 1$, thus giving slightly greater standard deviation values. After this correction, previously stated formula 2.1 gets following form:

$$s = \sqrt{\frac{\sum(x_i - \overline{X})^2}{n - 1}}.$$

Third, and most often used, way of describing the dispersion with SPC is to calculate the standard deviation of group means, also known as *standard error of means* (SE). By nature, SE has much tighter spread compared to standard deviation of individual samples. This is very useful feature when comparing process state during two time periods as subtle shifts in the mean value of distribution are not as easily concealed. SE is used also for calculating chart limits. Value for SE is calculated with $SE = \sigma/\sqrt{n}$, where σ equals standard deviation and n is the sample count. [4, pp. 92-93]

2.5 Chart limits

SPC can be useful only if it can detect problems within the process and alert operators to look for assignable causes. The most important detection methods are warning and action lines, which can be found in both mean and range charts. The meaning for each line types are that if calculated group average exceeds warning lines, it signals operators to monitor process closely while action line exceeds signal that there is practically no doubt that the process has gone out of control. Empirical research has shown that the best position in terms of detection sensitivity vs risk of false alarms is to position action lines at the distance of 3SE's (standard error

of means) from center line. If also alarm lines are used, they will be positioned at 2SE's (two-thirds of the action lines). This rule is also known as the three sigma limits and it holds for all chart types. It is also noteworthy to mention that the three sigma limit theorem is applicable to all distribution types, having only maximum of $2\% - 6\%$ change in detection sensitivity for extremely skewed distributions in comparison to normal distribution. [5, pp. 60, 65]

In practical SPC implementations, chart limits can be determined based on pre-calculated factors, which have been derived with statistical methods from subgroup averages and ranges. Although these factors don't take into account the skewness of the distribution, it has very little effect on the final limits as previously mentioned. The only parameters that matter are the subgroup size n and key parameters which can be calculated from sample data taken from the process. Depending on the used chart type, parameters like the grand average ($\overline{\overline{X}}$), average range ($\overline{R}$) and standard deviation (s) are commonly used. [5, p. 56] An example of one factor table for Range and Average chart type is shown in table 2.1 as an example.

Table 2.1 *Limit factors for Average and Range charts based on Avg Range, $\overline{R}$ [7, p. 419].*

n	A_2	D_3	D_4
2	1,880	–	3,268
3	1,023	–	2,574
4	0,729	–	2,282
5	0,577	–	2,114
6	0,483	–	2,004
7	0,419	0,076	1,924
8	0,373	0,136	1,864
9	0,337	0,184	1,846
10	0,308	0,223	1,777
⋮			

n = subgroup size

Formulas for control limits:

$$\begin{aligned} UCL_{\overline{X}} &= \overline{\overline{X}} + A_2\overline{R} \\ LCL_{\overline{X}} &= \overline{\overline{X}} - A_2\overline{R} \\ UCL_R &= D_4\overline{R} \\ LCL_R &= D_3\overline{R}, \end{aligned}$$

where $UCL_{\overline{X}}$ and $LCL_{\overline{X}}$ are the upper and lower action limits for Average chart respectively and UCL_R and LCL_R are the limits for Range chart. The factors A_2, D_3, and D_4 can be looked up from Table 2.1 when sample size n is known. [7, p. 419]

2.6 Chart types and interpretation

Monitoring of the process capability and state of control can be achieved with the help of various control charts. Several different control charts exist for continuous data, like Xbar-R, Xbar-S and I-MR. Selecting suitable chart depends on the process and available data. For example, I-MR chart is used when measurements are not grouped (group size of one), Xbar-R ('R' standing for range) chart is for cases where subgroups size is less or equal to eight and Xbar-S ('S' standing for standard deviation) is when sub group size exceeds eight. [6] An example of an Xbar-R chart is illustrated in Figure 2.7.

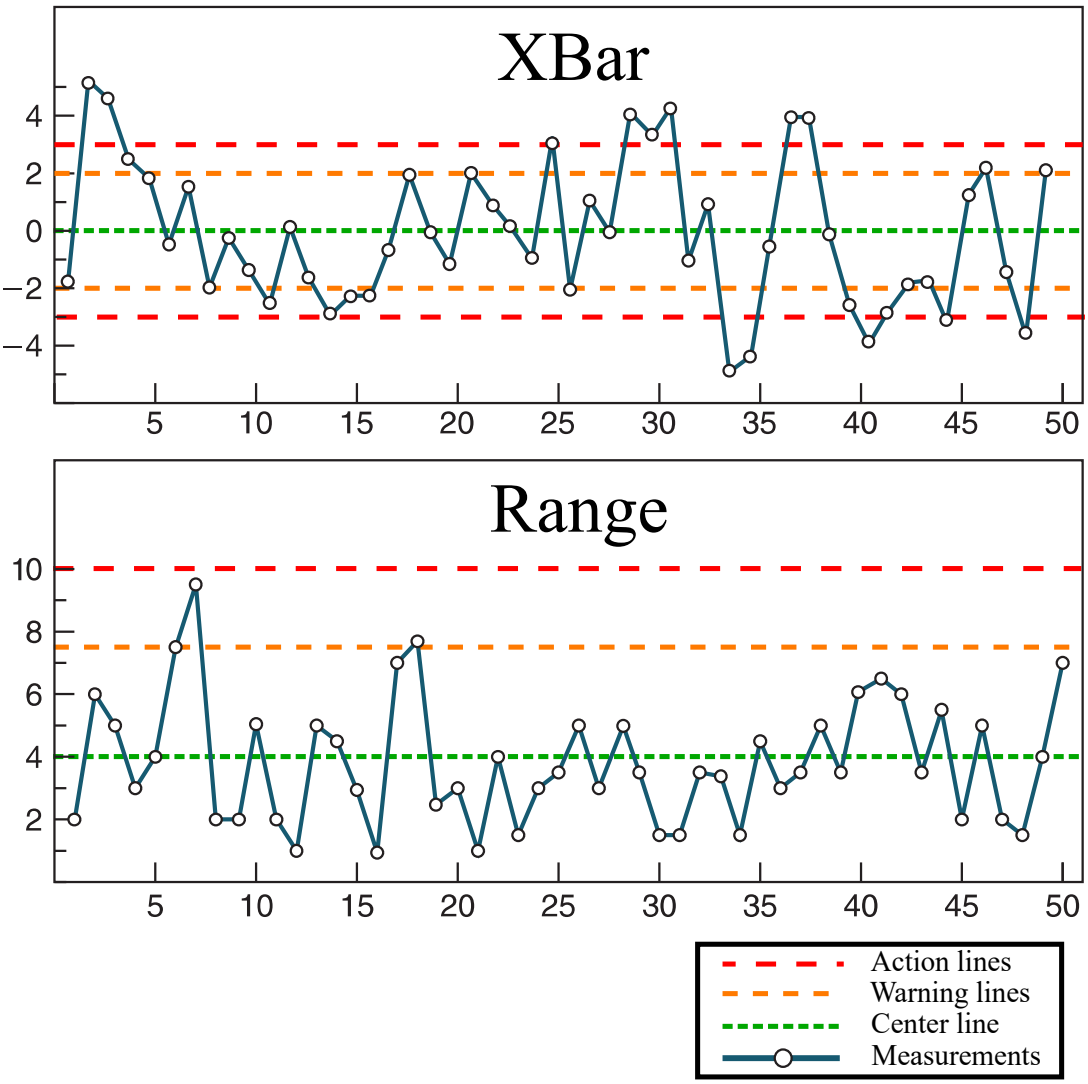


Figure 2.7 *Example of Xbar and Range charts. [4, p. 128]*

As can be seen in Figure 2.7, both Xbar and Range charts are plotted against time in x-axis, providing the possibility to visually examine the process' measurement history and development over time. Both charts also have two sets of limits, which provide traffic light signals for operators to check the state of control for the process. If values fall within warning lines ('green' area), process should be allowed to run without adjustments. If single value exceeds either of the warning lines ('yellow' area), it signals that the process should be monitored closely. Occasional exceeding of warning lines is, however, expected behavior, as warning lines are positioned at the distance of 2SE's, meaning that approximately one of every 40 measurements exceed

this line while process is in control. However, having two consequent measurements exceeding warning lines happens at a probability of $(1/40)^2 = 1/1600$ under normal conditions, indicating with almost full certainty that the process is out of control. Measurements exceeding action lines at the distance of 3SE's ('red' area) are in practice a clear indication that the process has moved to out of control state as the positioning of action lines gives the probability of exceeds approximately 1/1000. [4, pp. 109-110] [6] In addition to limit exceeds, other signs can be detected in control charts, which indicate a need to intervene in the process. These will be addressed in following section 2.7.

Another commonly used chart type is called CUSUM (Cumulative Sum) chart, which utilizes the the process history for detecting gradients or slopes within the process. The key idea behind CUSUM chart is to select or calculate a baseline value, typically denoted with μ_0, subtract all measurements from this value and run a continuous summation for the subtractions. The running summation value is generally called Cusum Score (Sr). An example of CUSUM chart is shown in Figure 2.8.

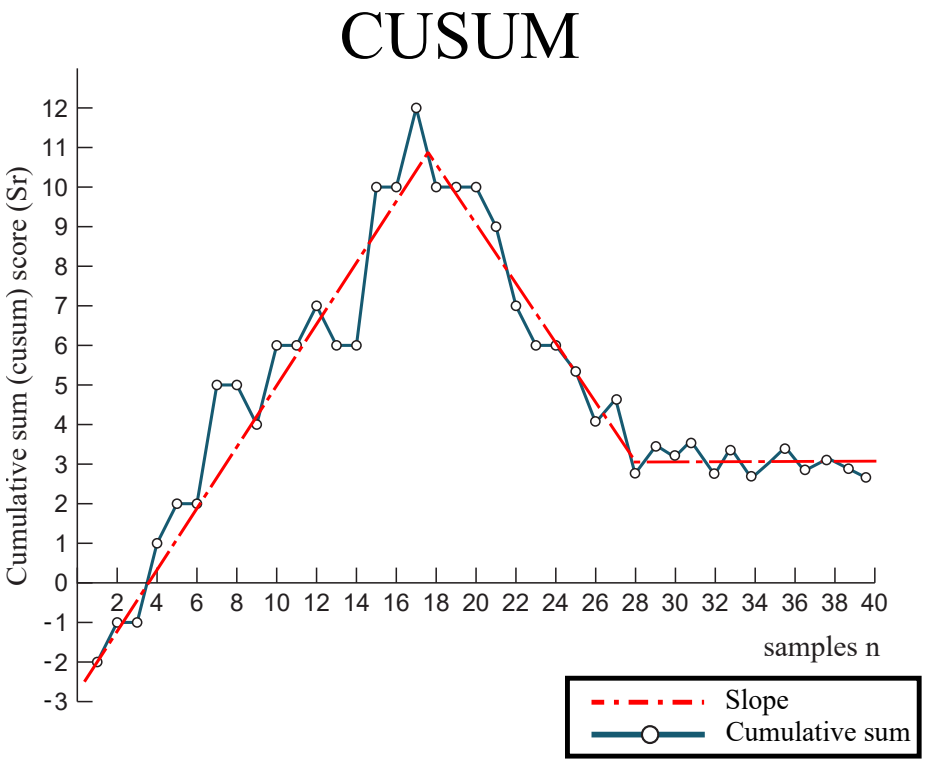


Figure 2.8 *Example CUSUM chart with two detectable slopes. [4, p. 226] [7, p. 294]*

As Figure 2.8 demonstrates, there are two clearly distinguishable slopes in the data. These slopes have been caused because the process average has first shifted above the selected μ_0 value and at around 17th sample it has shifted below. Around the 28th sample the process average has moved close to the value of μ_0, meaning that the slope angle is holding at 0 degrees and the Cusum Score stays relatively unchanged. Although the Cusum Score is running above the zero line, this has in practice very little meaning since the most important parts of the chart are the slopes and their steepness. [7, pp. 289-297]

2.7 Detecting problems

Once data collection and control charts have been established, they provide a powerful tool for detecting problems in the process. Typically, assignable causes manifest themselves as trends and patterns, clearly distinguishable from the charts. It should also be noted that control charts don't usually reveal the source of the problem – only that it exists. This also means that thorough knowledge of the process and its operating procedures is essential and used in combination with control charts troubleshooting becomes possible. [4, p. 321]

How the problem presents itself varies for every process and all-around solution for each and every case is impossible to produce. However, some general categorization can be provided depending on whether and how problems affect process average and standard deviation. Undesirable changes in the process may lead to three possible scenarios: changes in mean value, changes in spread and changes in both. Furthermore, these changes can be divided to several subcategories depending on the existence of cyclicality in the changes. [4, pp. 321-322]

1. Change in process mean
   (a) sustained drift (step change(s) in mean value)
   (b) drift or trend - including cyclical (slowly changing mean)
   (c) frequent irregular shifts
2. Change in standard deviation
   (a) sustained changes
   (b) drift or trend - including cyclical

(c) frequent irregular changes

3. Irregular changes in both mean and standard deviation

Examples of how these effects are shown in the process, and in the Range and Mean control charts are show in Figure 2.9. It is worth to note that many of the previously mentioned changes don't show in both charts but only either one. For this reason it is important to always examine both charts.

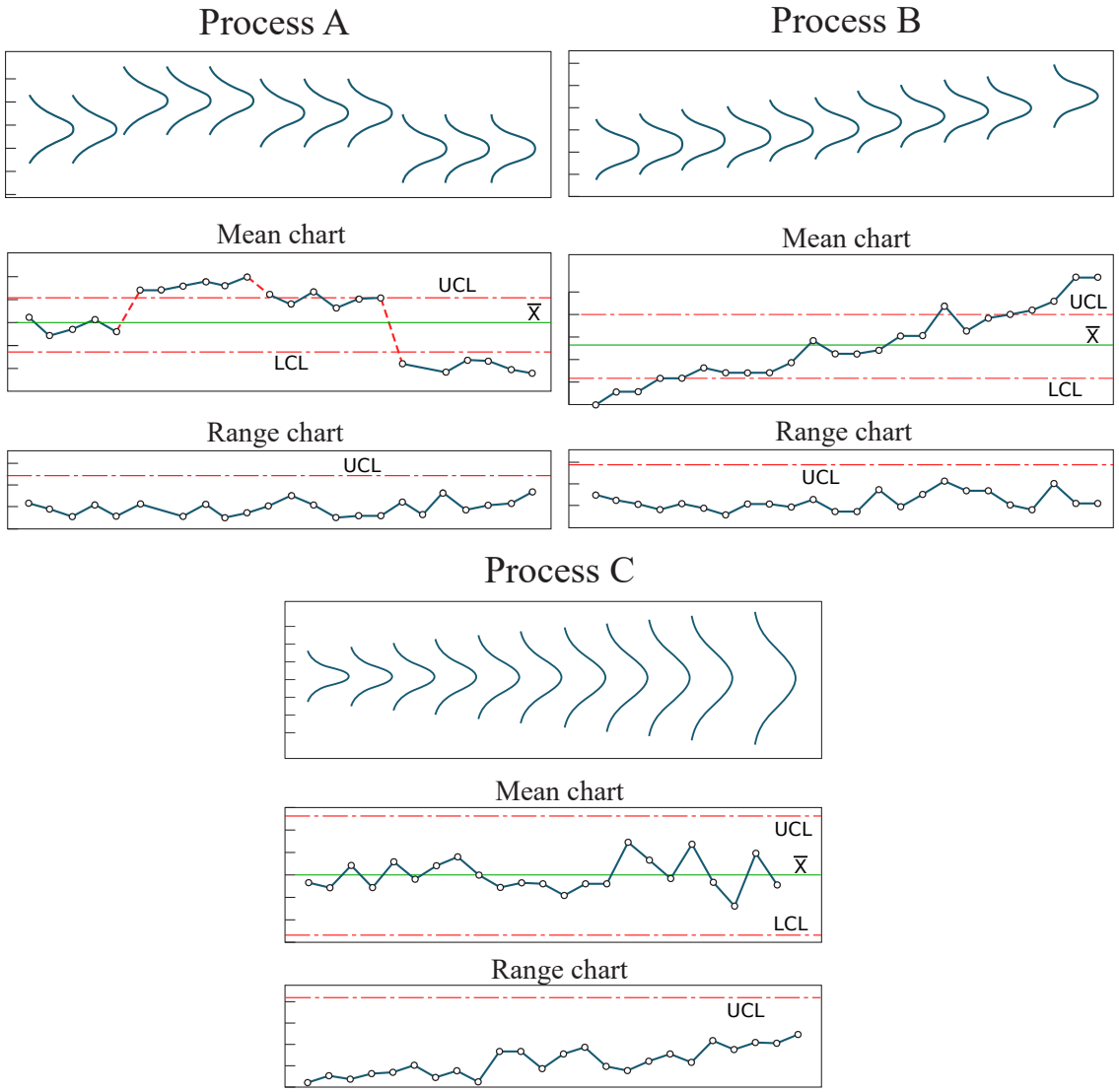


Figure 2.9 *Range and Mean charts showing different out-of-control situations [4, pp. 323, 324, 328].*

Figure 2.9 shows three different cases of common out of control situations. In Process A, the mean value has experienced two sudden shifts, first upwards and then downwards. As can be seen from corresponding control charts, the Range chart shows practically no signs of any problems. In the Mean chart, however, indications are clear as multiple consequent values run on either side of the center line. Some values have even crossed action limits, which would have been an indication of out of control situation alone. In Process B, consequent values drift from lower action limit towards upper action limit, forming a trend. This is also an indication of out of control situation. Once again, this effect is shown clearly only in the Mean chart. Finally, in Process C the mean value is holding its centrality, but its dispersion has started to gradually grow. In control charts this is clearly visible in the Range chart, where values have slowly started to climb towards the control limit line. The Mean chart shows little indications of any problems since in this case the process mean has not actually shifted. However, growing dispersion has started to affect the Mean chart also, since the offset from center line has grown in magnitude as time passes. [4, pp. 322-328]

Since the beginning of the computer era, it has become feasible to run automatic tests for monitoring process control state. Limit exceeds are the simplest ones to detect, but as show above, there are other kind of patterns for detecting out of control situations. One of the earliest of automatic detection tests are four Western Electric's rules originating from the 1920's. These have been later extended by Lloyd S. Nelson during the 1980's, containing eight rules in total:

1. One point is more than 3 standard deviations from the mean (outlier)
2. Nine (or more) points in a row are on the same side of the mean (shift)
3. Six (or more) points in a row are continually increasing (or decreasing) (trend)
4. Fourteen (or more) points in a row alternate in direction, increasing then decreasing (bimodal, 2 or more factors in data set)
5. Two (or three) out of three points in a row are more than 2 standard deviations from the mean in the same direction (shift)
6. Four (or five) out of five points in a row are more than 1 standard deviation from the mean in the same direction (shift or trend)

7. Fifteen points in a row are all within 1 standard deviation of the mean on either side of the mean (reduced variation or measurement issue)

8. Eight points in a row exist with none within 1 standard deviation of the mean and the points are in both directions from the mean (bimodal, 2 or more factors in data set)

In addition to having more rules compared to Western Electric's rules, Nelson rules have been adjusted so that their chances of detecting out of control situation are more evenly spread. However, whichever test set is used, it is important to comprehend that they are intended for guidance and alerting operator's attention instead of providing strict rules for determining out of control situations. Additionally, not all of the descibed rules may fit for every process and may be omitted at operators discretion in such cases where a lot of false alarms are produced by some of the tests. [8]

If there is a reasonable certainty to suspect that the process has gone out of control, the first thing to do is to look for any special causes. This happens by careful examination and break down of the complexity of the process. At this point it is essential that all important measurements are recorded, as well as adjustments that have been made to the process. By doing so it is possible to determine the causality between out of control process and changed process parameters. [4, p. 332]

3. SOFTWARE REQUIREMENTS

For any software development project it is essential to have a good understanding of what is the main purpose of the application and how it should interact with other systems. In case of Lab Entry, the application development has followed agile development principles, meaning that the software requirements have evolved throughout the project as better understanding has been acquired. Initially, the most important specification task was to uncover the core functionality of the application. This was done by studying DNALab feedback and feature requests collected throughout the years and by conducting several visits in customer laboratories. In terms of software development these features are called *functional requirements*. Another essential task was to find out what are the *non-functional requirements* for Lab Entry. In practice, this means in what environment application runs, how it interacts with other systems and what stability, availability and security demands are excepted. In order to have full understanding of how DNALab works and why certain design choices made there are also affecting Lab Entry, a look at Valmet DNA automation platform has to be made first. [9]

3.1 Valmet DNA

Valmet DNA is an automation and information platform for managing processes, machines, drives and quality controls. It is based on the knowledge of developing distributed control systems over 30 years and has been designed to meet requirements of high reliability, flexibility, as well as needs for sophisticated analyzing and reporting solutions. Main product categories in Valmet DNA are hardware controls and tools for engineering, maintenance, and operators. Engineering and maintenance tools are targeted for designing and managing automation loops with such applications as DNA Explorer and Function Block CAD. For operators, the most important tools are DNA Operate for real time monitoring and DNA Report for long term monitoring. Both DNA Operate and DNA Report make use of Valmet

DNA Historian database, which can store automation data for extended periods of time. [10] General structure of Valmet DNA Historian server is described in Figure 3.1.

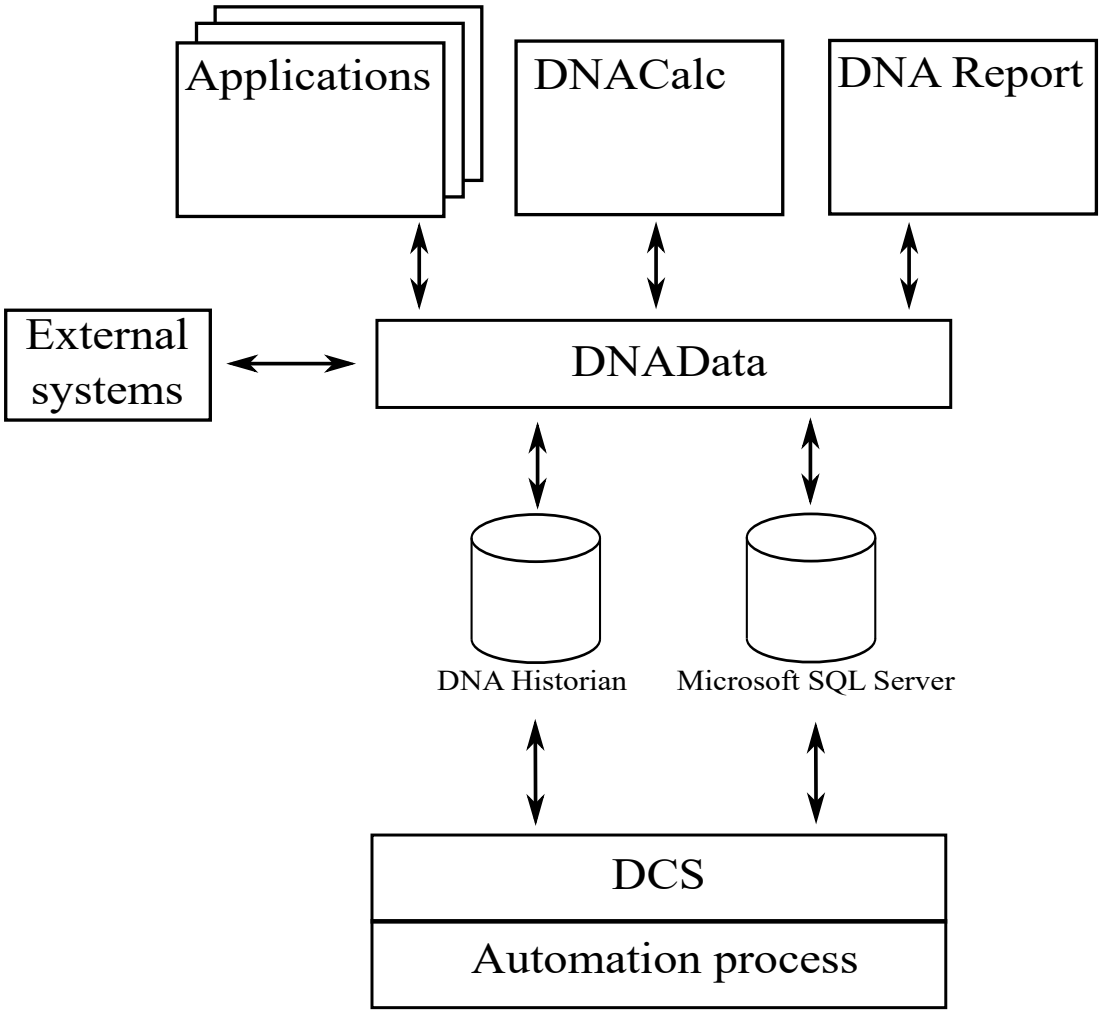


Figure 3.1 *General structure of Valmet DNA Reporting system with data flow.*

The most commonly used tools with DNA Historian are DNA Report Designer for reporting and DNACalc for executing complex calculations on collected automation data. With these tools it is possible to implement basic applications without having to be experienced in programming. [10] More complex applications are typically implemented with various techniques containing desktop software and database structures. Common aspects for every application apart from their complexity are platform structure, interfaces and databases. One of such interfaces is *DNAData*, which is an application programming interface (API) providing

access for applications to underlying databases and other services. Benefits for API utilization over direct database connections are standardization, security and backwards compatibility. DNAData plays an important part in inter-process data transfers, as well as communicating with 3rd party systems. As DNA Historian database and DNAData have an essential role in the development of Lab Entry application, their architectures are explained more detailed in following section 3.2. [11]

3.2 DNA Historian and DNAData

The main purpose for DNA Historian server is to collect and store real time data produced by the automation system. Each measurement or calculated value is called a *tag* and the total count for collected tags is typically measured in tens of thousands. Collection cycle can be 100ms at its shortest. DNA Historian gets its data from so called CIM-IO node, which acts as a buffer between automation network and DNA Historian server. Applications and external systems can access collected data by using DNAData interface. [12]

Most visible aspect of DNAData to application developer and end-user is its Web Services interface, which consists of standard interface description (WSDL and RDF) and SOAP protocol for method calls. [13] Most common operations are reading and writing to DNA Historian database, but it is also possible to expand the interface by writing new methods as needed. These DNAData methods are written with Visual Basic or C# and they may contain code for interfacing with other systems or performing complex calculations. In general, collections of these methods are called *DataClasses*. The most typical scenario for writing a custom DataClass is to have a data source for DNA Report. [14]

DataClasses have a good level of flexibility as it is possible to invoke other DataClass methods within a DataClass method, thus combining data from multiple sources. Additionally, new functionality may be implemented alongside existing products by hooking so called Trigger DataClass methods to existing methods, minimally interfering with target product. Also, multiple DNA Historian server databases can be configured so that data can be queried from external systems if needed. [14]

3.3 DNALab overview and feedback

After taking a general overview of the most important parts of Valmet DNA in relation to the development project, its time to take a closer look at the application that is being replaced, DNALab. As being developed since the early 2000's, DNALab has gone through a lot of improvements, many of them originating from customer feature requests and feedback. This information is essential in the development of Lab Entry as many functional requirements can be created based on the features that are found in DNALab. Additionally, examining the structure of DNALab has considerable effects on the non-functional requirements of Lab Entry as these two applications need to have some level of compatibility.

Basically, DNALab is a combination of two desktop applications: Entry and Conf. The main purpose for Entry application is to provide tools for every day laboratory work, whereas Conf is for administration and configuration purposes. In combination these applications provide useful features like categorizing analyses based on factory sections and sample places. In addition to just inserting and editing analysis values, DNALab contains so called calculation feature where user may define simple calculation formulas (for example $result = (A + B)/2$) to reduce reoccurring calculation tasks related to some analyses. Typically, laboratory analysis results are used to monitor and calibrate automatic analyzers. For this reason it is a common task to align laboratory analyses with one or more process measurements. Timestamp-based alignment is done automatically with DNALab and the subtraction between matched values is calculated in order to indicate more clearly any differences. [15] General structure for DNALab and its relation to different databases is presented in Figure 3.2

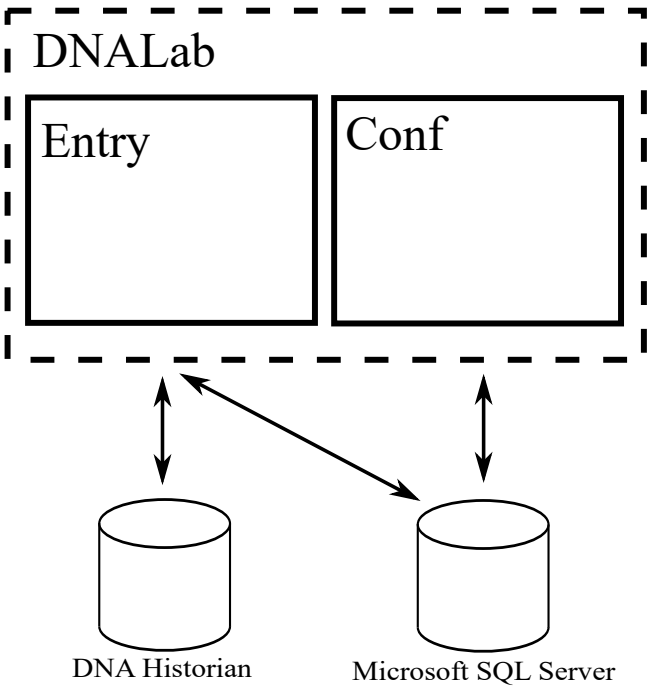


Figure 3.2 *DNALab application structure.*

Internally DNALab uses DNA Historian database for storing analysis values and Microsoft SQL Server for managing configuration and analysis metadata. Decision to use separate databases for analyses and configuration has been made due to the different use cases for each databases. As DNA Historian database is the standard solution for storing all real time automation data in Valmet DNA, it is a natural choice and eases access for other applications to analysis data through DNAData. However, DNA Historian doesn't handle well more complex relational data and because of this, all configuration values are saved in Microsoft SQL server instead. It is also notable to mention that DNALab uses direct ODBC (Open Database Connection) to access DNA Historian instead of DNAData. This is due to the fact that DNALab predates DNAData by several years. [15]

During its many years of active use by many customers, DNALab has collected a lot of feedback. Some of that is bug reports, which generally don't concern Lab Entry. However, part of the feedback is related to more general ideas towards laboratory work and could potentially lead to adopting new working methods and more efficient and reliable usage of laboratory entry system.

When examining the feedback data for DNALab, the most commonly occurring theme was that analysis hierarchy should be more like daily workflow instead of factory section hierarchy. Additionally, some factories are so large that they have multiple laboratories, which all should have their own hierarchies in order to find correct analyses easily. Also, SPC features were commonly requested as DNALab doesn't currently contain them in itself and they are implemented with separate application based on DNA Report. Navigating between multiple applications was found to be be tedious and time consuming. [16]

3.4 Customer interviews

Initial customer visits were conducted during early phases of the project with main objectives on gathering information about laboratory work: what tasks were frequent, what was done similarly and what differently. Also, any targets for improvement were noted so that Lab Entry would be designed to be the most helpful in these tasks. In total three initial customer visits were done. Target sites for these two interviews were both pulp mills, later addressed as *Pulp mill 1* and *Pulp mill 2*. Third visit was done to a Fabrics factory with the intent to have wider perspective to laboratory work in addition to pulp mills. Summaries for all visits can be found in appendixes 1, 2 and 3.

Common findings for both pulp mill interviews were that work in laboratory was generally planned based on weekly or monthly schedule, but exceptions were common due to process state, equipment malfunctions or sick leaves. Thus it was important to react to changing situations flexibly. In addition to just making analyses, daily routines included many supportive tasks, like cleaning equipment and ordering chemicals. These were commonly marked in calendar along analyses. Usually work was done in self-organizing manner where tasks were done by whomever was available. Most of the communication happened verbally.

The most notable differences were that the laboratory team in Pulp mill 1 relied more on excel-based calculations and SPC charts on determining whether process is in control, while Pulp mill 2 team used more intuitive approach based on simple alarm limits only. Also, reporting about analysis results happened mainly by phone in Pulp mill 2, while Pulp mill 1 relied more on emailing results – some of them automatically based on analysis values crossing limits or SPC falling to out of control state.

Several improvement targets were also found. For example, analysis results were inserted to several places, like personal notebooks, excel sheets, DNALab and email messages for interested people. Ideally, all these tasks could be combined under single application with the need of inserting value only once. Additionally, it was noted that while some analysis results were obtained directly from the analyzer instrument, others needed to be calculated with a pocket calculator from the initial values. A common example of such calculation was density, which was calculated from subtracting empty bottle weight from full bottle weight (fluid weight) and dividing it with fluid volume.

Perhaps the most complicated manual calculation task was done when determining the concentration of Sodium Hydroxide (NaOH), which is a component in *white liquor* used for separating lignin from cellulose. Steps done in this analysis included cooling down the fluid to specific temperature (25 °C), calculating density as described previously and using a lookup table for concentration once both values were known. Clearly this manual task could be automated with a function which interpolates concentration from array of values with given temperature and density.

3.5 Functional requirements

Based on customer interview results, feedback, and internal meeting notes, Lab Entry's functional requirements were created by first writing down formalized use cases of what actions user should be able to perform. Due to the fact that the complete list of use cases was found to be quite extensive, application's implementation was divided to smaller sections. This also lead to the decision that not all features were planned to be implemented in the first version but later on as system development advances further. Some use cases were clearly related to configuration, which will be managed through DNALab Conf in the first version of the application. Furthermore, related use cases were grouped as features, like *"Analysis Entry"* for example, which in addition to just adding new entries includes related tasks like deleting and editing existing entries. Most central feature requests with their planned implementation phase have been collected to Table 3.1.

Table 3.1 *Lab Entry feature requests and planned implementation phase.*

Feature	DNALab	Pulp mill 1	Pulp mill 2	Fabrics	Phase
Analysis searching	✓	✓	✓	✓	Internal pilot
Analysis entry	✓	✓	✓	✓	Internal pilot
Commenting	✓	✓	✓	✓	Internal pilot
Proc.meas. align	✓	✓	✓		Internal pilot
SPC Charts		✓	✓	✓	Internal pilot
Calculator	✓[1]	✓	?	?	Customer pilot
Weekprogram		✓	?		Customer pilot
Collection samples		✓	✓	✓	Final product
Batch numbers				✓	No plans
File attachments				✓	No plans

[1] Limited functionality.

? Might be partially useful.

In Table 3.1 the most central features have been listed from top-down based on planned implementation timeline. The most decisive factor when determining implementation order was dependencies to other features. For example, SPC charts are no use on their own if analysis entry or process measurement alignment features are not implemented first. Implementation was divided in three phases, where first Internal pilot was conducted by assigned test team. During customer pilot all core features for daily laboratory work were implemented. Third phase, the final product, contains features which have been agreed on to be included in final product, but are not critical to every day work. Finally, at the bottom of the table are features that have been acknowledged, but are either too complex to implement or not regarded as important enough.

3.6 Non-functional requirements

In comparison to functional requirements, every software has so called non-functional requirements, which contain implicit expectations of how well software should work. Sometimes also called as 'software quality attributes', non-functional requirements define features like availability, efficiency, reliability, security and robustness.

Unlike functional requirements, none of the non-functional requirements change the behavior of the software. [17, pp. 113-114] This section contains the most important non-functional requirements, which have to be taken into account from the beginning of the development work.

3.6.1 Interfaces and compatibility

In order to keep the amount of required development work reasonable, it has been planned to use existing DNALab Conf application with Lab Entry. This design choice also means that Lab Entry has to be largely compatible with DNALab Conf's database structures. In practice, this requires Lab Entry to use Microsoft SQL database as its primary means of storing configuration data. Additionally, some new configurations will be needed in order to manage completely new features like week program. These new datatable structures will get added alongside existing configuration, altering DNALab's database tables as little as possible. This way it is possible to ensure that DNALab Conf doesn't cease to function due to database changes.

In addition to storing configuration data persistently, analysis data will also require a database. Storing and sharing the analysis entries among other applications happens most easily through DNAData interface, which in turn uses DNA Historian database. For this reason Lab Entry needs to have means of interfacing with DNAData. Another benefit for using DNA Historian is fast execution for numeric operations, like calculating long-term averages, standard deviations, minimum and maximum values, as these are highly optimized features.

3.6.2 Quality and performance requirements

For quality related issues it is a common fact that the costs increase exponentially the later issues are detected. As the Lab Entry application will be handling production critical data, it is the top priority to achieve high availability and data persistence so that no data will get lost due to incorrect behavior or unavailability of the system. In order to guarantee both of these requirements, software quality management has to be planned carefully early on. Some aspects of application quality are to be taken into account during specification and design phase while others, like testing, validation and verification come later. Following elements have been brought up

as they have quality improving properties, which will overcome later fateful design flaws in the application design.

Concerning data persistence matter, it has been planned that user interactions, like new analysis values, edits and comments for example, are stored to several independent data storages in case any of them malfunctions at any given moment. In practice, this would mean utilizing both DNA Historian database, as well as Microsoft SQL server for storing data. As a last option for losing all database connections at once, all interactions are written directly to a file system log for later recovery.

If, despite all preventive measures, some unforeseen fault occurs, leaving system unusable or causing random errors, extensive diagnostic features need to be established. In practice this can be done by logging critical errors and showing clear error messages to system users, indicating that there were problems with the performed operation. Additionally, some problems may cause system to slow down over time, leading to decreased user experience. Such problem could occur, for example, in a case where a database query slows down due to missing indexes. In order to detect such performance issues, performance counters will be implemented to any operations which affect system responsiveness and speed.

4. WEB DEVELOPMENT

With requirements laid down in previous chapter, the next task in the development process was to decide what technologies will be selected for the application. Initial choice between traditional desktop application and more modern web application was resolved in favor of web application, due to the benefits that web applications have over desktop applications. To name a few, such features as platform independence, easy client installation and delivery of updates, concurrent client support and good availability of development tools were considered beneficial. Even after making the decision of using web techniques, there remains a substantial amount of different design patterns, frameworks and libraries to choose from, each providing tools for different type of applications. This chapter makes a general overview to web applications, their development techniques and common security challenges which concern every web application.

4.1 Basic concepts

Web application can be defined as an application, which uses a web browser as its client platform. Client is associated with a server, which serves the client application to user's web browser and handles requests sent by the client, for example saving user inputs to a database. Client applications have several different implementation techniques, for example Java and Adobe Flash, which both have suffered decrease in popularity during recent years. These techniques have been replaced by the combination of HTML, JavaScript and CSS. [18] On the server-side implementation techniques have more variation, but the winner stands out clearly as PHP has managed to hold its position as the most popular programming language with roughly 83% share of known web sites while ASP.NET is at 2nd place with 14% share according to W3Techs survey [19].

Between the client and the server stands an application programming interface, which provides a surface to clients for interacting with the server. Depending on

the implementation, client can access different resources by altering the request URL, HTTP verbs, route parameters or the contents of the HTTP call header and body. In addition to this, the server may respond with HTML, XML, JSON, file or with plain status code. In order to manage the multitude of available request and response options, several design architectures have been developed for managing interoperability between different systems. One such architecture is called Web Services, which was briefly mentioned in the section 3.2 as being the standard interface for accessing DNAData methods. Nowadays, however, other API architectures have taken its place where REST (REpresentational State Transfer) being one commonly used. The key principles behind REST are distinction between resources and their representations, statelessness, and links in response leading to other resources. [20]

Web application's server infrastructure can vary depending on the application type, expected user counts and requirements for availability, cost, scalability and ease of management. In the simplest infrastructure only single server is used for both as an application and database server. Although being simple, this approach doesn't scale well, and leaves the server application and database to compete on the same server resources (CPU, RAM, I/O). A better approach in the sense of performance and scalability is to separate application and database servers from each other. This way it is easier to determine bottlenecks in the system and increase resources where needed. [21]

Further scaling up the system can be done by adding parallel application servers, increasing the capacity for concurrent users. When having more than one application server, a new requirement rises for distributing client requests among application servers. This can be achieved with a load balancer, whose main task is to distribute the workload among application servers. Using load balancer also increases availability as failure in one application server gets distributed among other servers until the failed server has been fixed. Database can also be scaled up by adding more databases to the system, where one database acts as a master database and the rest as slaves. The role for the master database is to perform read and write operations while other databases handle only read operations. Data updates are managed by performing replication operations directly from the master database to its slaves. The main benefit for using master-slave approach is in such case where read operations are performed more often than writes. [21]

Previously described infrastructure choices can be implemented with physical routers, servers and firewalls, but for some time it has been common to use virtualization for achieving similar results. Currently multiple companies, like Microsoft Azure, Digital Ocean and Amazon Web Services, provide such virtual platforms, better known as clouds. Cloud services can be divided to three tiers, SaaS, PaaS, and IaaS, depending on the end-user's capabilities and responsibilities on the platform. The most basic way of utilizing cloud services is the SaaS (Software as a Service) level, where user buys an application as a service instead of more traditional licensing model. Access to application happens most commonly with a browser. Second level, PaaS (Platform as a Service), provides user a platform for running own applications or web sites. Final level, IaaS (Infrastructure as a Service) provides end-user the greatest freedom of all three with the possibility to commission new virtual machines, configure firewalls and virtual networks. An example of a web application in SaaS-level infrastructure with three application servers, load balancer and two databases is shown in Figure 4.1. [22]

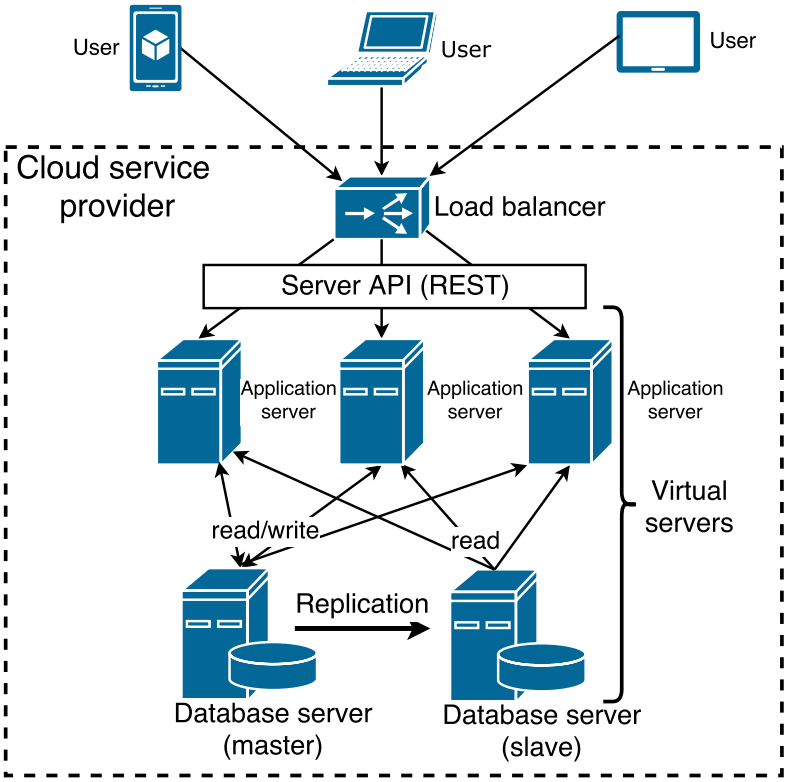


Figure 4.1 *Web application infrastructure in a cloud environment. [21]*

In addition to physical (or virtual) hardware, web server needs an application for responding to client requests. Typically, the server application is accessed through a domain name (like *example.com*), while resources in the HTTP server are identified by URL's (Universal Resource Locator). URL's consist of the domain part and route part pointing to a specific resource in the server. Resources in the HTTP server may, for example, be static files, rendered upon requests, or data queried from a database. HTTP servers also manage user authentication as well as determining which resources user is allowed to access. [23] [24]

As mentioned earlier, client applications are nowadays almost exclusively implemented with HTML, CSS and JavaScript. The role of HTML (Hypertext Markup Language) is to provide the basic structure for web pages. CSS (Cascading Style Sheets) gives pages their styling, for example colors and fonts, as well as how HTML elements should be aligned in relation to each other. The purpose of JavaScript is to provide functionality for the page. This includes handling user inputs, showing messages and animations, as well as communicating with the server and manipulating the structure of the web page, also known as DOM (Document Object Model). [25] In order to manage the complexity of modern web applications, several design patterns, frameworks and libraries have been developed. Selecting them correctly can have a profounding impact on development time, amount of bugs and maintainability. These topics will be discussed in the following sections.

4.2 Design patterns

One of the key factors for a successful software project is to choose suitable design patterns which fit to the problem at hand. This way it is possible to guide codebase structure to a more maintainable form and prevent minor issues which can later lead to major problems. Basically, a design pattern is a reusable solution which can be applied to commonly occurring problems in software design. Design patterns are not exact solutions to any specific problem, but they provide generic guidance and structures how problems should be solved. Typically, successful design patterns have been developed over long periods of time and they have proven their worth many times over before gaining acceptance among developer community. [26]. Below is a general overview of some of the most popular design patterns used in web development.

4.2.1 Model–View–Controller

Model–View–Controller (MVC) is an architectural design pattern which encourages application structuring through the separation of concerns. MVC has its roots in late 1970's where it was initially used to improve code reuseability throughout application by decoupling user interface and application logic from each other. Nowadays, MVC pattern is supported by wide range of different type of programming languages, including JavaScript. Though having endured throughout the years, same three key elements, model, view and controller, can still be found in the pattern. [27] Relations between the three concerns are displayed in Figure 4.2.

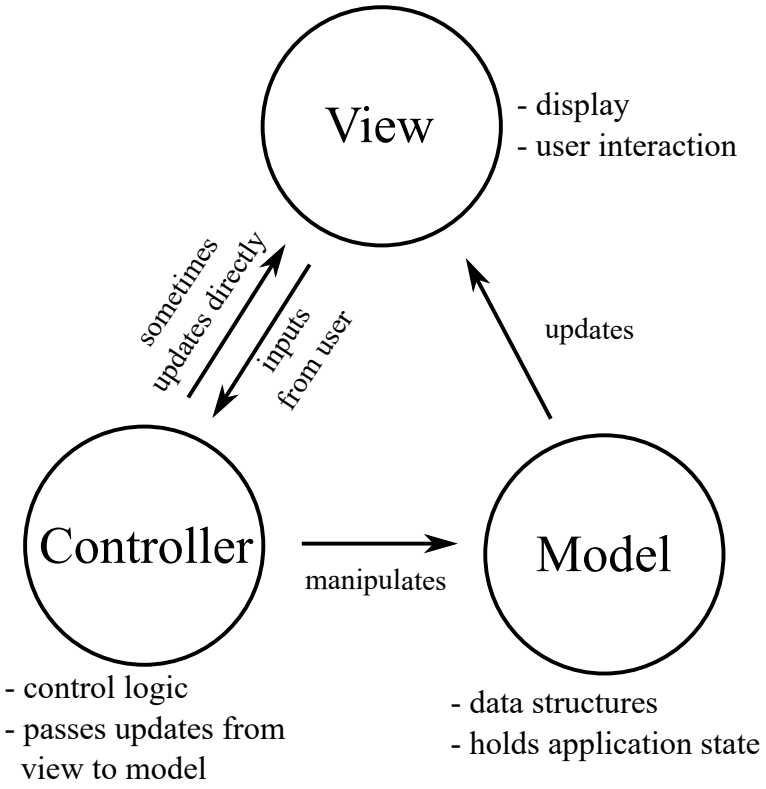


Figure 4.2 *General overview of concerns and their interaction in MVC pattern. [27].*

The main purpose for the model part in MVC is to hold application's data and provide interface for updating it. Usually, model also notifies its observers about changes in its state. In practical applications, models typically validate their attributes so that their integrity is not compromised. If user data needs to be saved between sessions, model is the one concern to handle data persistence. In context

of web applications this could be either memory, localStorage or database. In some applications multiple models are grouped together with the benefit of avoiding the need to observe each model individually and use common interface instead. For example, JavaScript library *Backbone.js* supports this kind of behavior [28]. [26, p. 116]

View's responsibility in MVC pattern is the representation of model's data in a user-friendly manner, applying filtering or ordering if needed. Additionally, view takes user inputs and updates model accordingly. Any updates on model are reflected to views through various notification mechanisms, like Observer pattern. Even though user inputs to view are reflected to model, the actual task of handling the updates is assigned to Controller. Like in the case of model, multiple JavaScript libraries can be found for implementing views, like *Lodash* library's templating engine for DOM manipulation [29]. [26, p. 116]

The third concern, Controller, acts as an intermediary between View and Model. In other words, user input data flows to model through the Controller, which has better knowledge how the Model updates should be handled. This renders the View less dependent on Model's implementation details. Additionally, Controller also handles data formatting, like ordering lists alphabetically. In this case no Model interaction is needed but the Controller updates the View directly. For many client-side JavaScript library implementations it is a common practice to take a non-traditional approach for the role of the Controller. Reasons for this vary, but the most common arguments are related to the fact that the traditional role of the Controller doesn't translate strictly one-for-one to client-side applications. [26, p. 121]

4.2.2 Module pattern

One profounding problem with JavaScript is the lacking concept of classes. Classes are used for encapsulating functionality behind public interface methods, shielding object's internal state from undesired or unintentional modifications. Additionally, declaring some methods and variables private avoids exposing them to the global scope, which could potentially lead to introducing difficultly solvable bugs in the program. For these reasons practically every non-trivial web application needs to adopt some sort of module pattern for separating different code blocks from each other. Even if the application is implemented with module pattern, it is important

for developer to comprehend that every module pattern implementation merely emulates classes, since achieving truly private variables and methods is not possible in JavaScript. Privacy can though be effectively achieved by using the function scope, which causes variables declared inside functions to be visible only within that scope. [26, pp. 26-28]

Many JavaScript frameworks implement module pattern, like AMD modules, CommonJS and SystemJS. One popular library implementation for managing AMD modules is called RequireJS, whose main purpose is to load JavaScript modules, as well as figure out in which order modules have to be loaded based on their dependencies. From the web application developer perspective, using RequireJS reduces greatly the time it takes to integrate external libraries to project and significantly enhances debugging capabilities, should there be any problems with missing dependencies. Other benefits of using RequireJS are asynchronous module loading, which increases performance, and tools for compressing source codes, effectively reducing the size of the files loaded from the server. [30]

4.2.3 Dependency injection

A common way for application components to access other components and resources is to instantiate and build them as needed. This, however, leads to inflexibility and unnecessary dependencies between components, as the initiating component needs to 'know' what information child component needs. Such inflexibility manifests itself when the implementation of child component changes, leading to necessary changes in parent component. [31]

One way of achieving flexibility is to use so called dependency injection (DI), which in practice means that the child component is provided readily initialized as a parameter in the parent component's constructor. This way the parent doesn't need to know about the details of child component or its initialization. This mechanism also simplifies testing as the child component may be easily replaced to a test version as long as its interface remains the same. Although dependency injection is not completely new idea, it has become best known from Angular framework, where the whole application structure is based on using the dependency injection pattern. [31]

4.2.4 Publish–subscribe pattern

The publish–subscribe pattern is targeted for managing information flow within the application by sending events between different components. Depending on the role of the object in the pattern, it may either be subscriber, meaning that it is listening for events, or publisher, meaning that it acts as the sender of events. In comparison to more simple Observer pattern, where listeners attach directly to event source (the subject), in publish–subscribe pattern events are passed through an event channel. In the context of JavaScript, publish–subscribe pattern works well as in the browser most tasks are executed based on events, which have been initialized as user interacts with the web page. [26, pp. 50-52] An illustration of the publish–subscribe pattern between one server and several clients is shown in Figure 4.3.

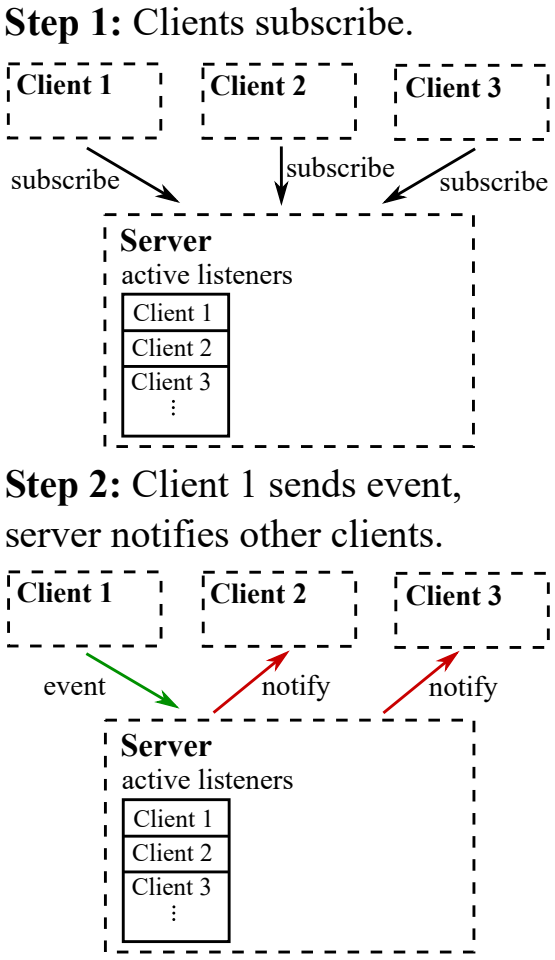


Figure 4.3 *Publish–subscribe pattern between clients and the server.*

Benefit for using the publish–subscribe or simpler observer pattern is the fact that it encourages the user to think of the roles different components have in the system. This could contribute in breaking down the system to smaller, more maintainable, blocks which are more loosely dependent on each other. Another advantage is the possibility to dynamically manage component relationships, meaning that the amount of publishers and subscribers, as well as their targets, can vary depending on the application's state. On the down-side of either pattern, is that it can sometimes be difficult to guarantee that the application as a whole behaves as expected. For example, the publisher may expect that at least one subscriber is always present, for instance handling and logging error situations in the publisher, but this could be problematic if the subscriber itself crashes or malfunctions. [26, pp. 46-52]

4.3 Frameworks and libraries

Where previously introduced design patterns are purely conceptual ideas how things should be done, frameworks are actual code structures helping to implement applications with chosen design patterns. For example, if it has been decided that application should take advantage of the MVC pattern, Angular is a good choice as a framework, since it naturally enforces using of MVC. [32] However, choosing correct framework (or none at all) is not an easy task and one should ask several questions about the application: what is its main purpose? What design patterns are used and how well framework supports them? What conditions are set by the platform? [33]

In addition to frameworks, applications may use libraries, which increase the abstraction level by encapsulating complex operations behind interfaces. Libraries do not impose any design patters as frameworks do, so they are much easier to adopt to projects during development. One example of a very popular front end library is *jQuery* [34], whose main purpose is to ease DOM traversing and element manipulation by using similar syntax to CSS selectors. Another great library for developing modular JavaScript is called Backbone.js [28], which emulates inheritance – a common concept from object-oriented programming.

4.4 Security

Since most web applications are typically exposed directly to internet or intranet, careful security measures have to be established in order to guarantee the safety of the application and sensitive user information. Although most automation networks are strictly separated from internet by multiple layers of firewalls and subnetworks, any potential security vulnerabilities should not be overlooked in case of network breach or users with harmful intentions. This section contains some of the most central security aspects which concern almost every web application and some guidelines how they should be accounted during application development. A list of top 10 web application attack types is maintained by OWASP (Open Web Application Security Project) and top three of them have been introduced below. [35]

4.4.1 Injection

Injection is a form of attack where hacker tries to inject code syntax via user input fields in order to affect targeted code interpreter. This is very typical case with database queries, SQL injection being a common example. If injection attack is successful, attacker may either steal, modify or delete database data all together. In many studies injection is regarded as the most harmful type of attack. [35]

In order to protect applications from injection attacks, both application and database driver developers should recognize potential risks and know how to produce safe code which is impervious to injection attacks. Typically, this is done by parametrisizing database queries and escaping all user inputs so that they won't be interpreted as part of the query. Most database drivers, like Tedious for Microsoft SQL, enforce escaping automatically. [36] [37]

4.4.2 Cross-site scripting

Cross-Site scripting (XSS) is regarded as the most common attack types and can be categorized to two forms: reflected and stored attacks. Reflected (or sometimes non-persistent) XSS attacks are typically in a form of a link to a website where malicious code is inserted along with HTTP request parameters. The actual

vulnerability occurs in such case where parameters are reflected back to user and then executed in his/her browser. The mechanism for how this attack type can work is due to the fact that the injected code, also known as attack vector, appears to be coming from a 'trusted' server. The most serious consequence of a successful XSS attack is a case where hacker gains access to victim user's session cookie and may then impersonate as him/her until the cookie expires. [35]

The seconds and more severe type of XSS is persistent attack, which means that attacker has successfully injected malicious HTML code to the target server, where it is served to all clients. Usually this kind of injection mean serious design flaws in the server or database. A simple example of persistent attack is a chat forum where user input is not sanitized and appropriately formed JavaScript code input gets delivered to all clients, infecting their session. [38]

The best way to avoid XSS attacks is to follow certain design rules where potentially untrusted content is either sanitized by removing illegal characters or escaping them so that they won't be interpreted as executable JavaScript code. Examples of common places for inserting such untrusted content are inside HTML elements, element properties, script tags and even HTML comment tags. Although good design practices are irreplaceable, there are some libraries specially designed for completing input sanitation tasks, which may provide some level of security. [39]

4.4.3 Broken authentication and session management

Building custom authentication schemes for applications is common practice, but doing it correctly is often difficult. Flaws may occur in areas like logout, password management, timeouts and 'remember me' or 'password recovery' -like features. Finding these flaws may prove to be a difficult task due to the uniqueness of each implementation. As a result of successful attack external (or internal) user may act as the victim user. Depending on the victim user's role in the system this may have effects on sensitive business data or even on application's behavior if administrator permissions have been acquired. [40]

Shielding system against broken authentication and session management can be best achieved by following good development practices. In general, application should have a single, well documented set of session management controls, which have simple, easy to understand interface for developers. [40] In the context of Node.js

multiple frameworks exist for managing user authentication and access to application operations. For example, such frameworks as Passport.js [41] and JSON Web Token [42] have gained a lot of popularity in Node.js community.

5. TECHNOLOGY SELECTIONS

Previous chapter made a general overview of commonly used web development techniques without going too much into details. This chapter takes a more thorough look at the technology selections that have been made with Lab Entry. The main focus is on the features described in chapter 3 and what qualities a selected technology needs to satisfy in order to contribute to building a fully functioning and easily maintainable application. It is important to remember that not only selected technologies have to be fit for the task, but also be compatible with each other as well.

5.1 Front end

Front end technology selections comprise exclusively of client-side HTML, CSS and JavaScript libraries. The most important features required from considered libraries were to provide tools for structuring code, interacting with server and web page's DOM elements. Some of the libraries have support for both AMD and NPM module loading, meaning that they can be used on both client and Node.js server environment. These were favored over those which could be used only on client side.

5.1.1 Structure

The most important means of structuring JavaScript code blocks in the Lab Entry application is the separation of functionality to smaller modules in such way that is described in previous *Module Pattern* section 4.2.2. In practice, module pattern has been implemented by following the Asynchronous Module Definition (AMD) mechanism. The main idea behind AMD is to overcome the problem of manually managing JavaScript dependencies and codeblock loading order with script tags, which may lead to introducing difficult bugs in the program if not done carefully. The AMD also handles dependencies to other modules, meaning that the application

code blocks won't get executed before all their dependencies have been loaded. In order to make use of AMD modules effectively, a library called RequireJS was utilized for the task of loading modules. [30] [43]

Another structural framework used in combination with RequireJS is Backbone.js, whose main purpose is to bind model data to corresponding views. Additionally, Backbone.js includes a model extension feature, meaning that any model can be extended to other models, inheriting base models properties and methods. In practice, this functionality has been utilized for developing *ComponentBase* model, which can be easily extended to other components. ComponentBase model contains properties and functions used by all components, for example functions for server communication, configuration management and mechanism for sending events between components. [28]

5.1.2 Table elements

Based on the software requirements specification it was recognized that there was a demand for a general purpose table component for displaying analyses and entry history with SPC key values and automated test results. HTML contains in itself a table element type, but its capabilities are limited to very basic form of displaying data. Such qualities as paging, searching based on meta data, row ordering, and dynamic updating were all on the list of required features which needed to be satisfied.

After exploring several available options, including developing the table component itself from the very beginning, a jQuery table plug-in named *Datatables* was chosen. To its benefit, Datatables provides extensive sorting and filtering options and its data source may vary from static DOM tree to a JSON source queried from server. Latter option was found convenient for handling changes in data source, should any of Lab Entry's users update data in the server. Additionally, Datatables comes with the option for writing a custom renderer methods for table cells, meaning that such elements as date pickers, comment input fields and edit buttons were possible to be implement. [44]

5.1.3 SPC chart component

Since statistical process control relies heavily on visualizing process state with charts, it is extremely important to implement required charts properly. For this reason a lot of effort was focused on researching possible chart libraries with their strengths and weaknesses. A list of top five chart components and requirements set by Lab Entry are listed in Table 5.1.

Table 5.1 *Lab Entry chart library comparison and requirements.*

Library	**D3.js**	**dc.js**	**Chart.js**	**Google Charts**	**Canvas.js**
Technology	SVG	SVG	HTML5	HTML5/SVG	HTML5
License	BSD	Apache2	MIT	Creat.Comm.	paid
Chart types	+100	+40	8	+18	30
Max data points	<50k	16k	<100k	≪30k	+50k
GitHub projects	7,7k	<400	2,7k	1,6k	2,4k
Stack Overflow tags	60k	2,6k	10k	13,6k	200
Trendline	✓	✓	✓	✓	✓
Zooming	✓	✓	✓*	✓	✓
Panning	✓	✓	✓*	✓	✓
Dynamic updating	✓	✓	✓	✓	✓
Empty datapoints	✓	✓	✓	✓	✓
Highlight datapoints	✓	✓*	✓	✓	-
Multitrend	✓	✓	✓	✓	✓
Tooltips	✓	✓	✓	✓	✓

* = Requires scripting or using plugins.

First observation while researching chart libraries was that there are plentifully options available for developing charts for HTML applications and the selected five in Table 5.1 represent only small section of all explored options. It was also found that many of the available libraries were able to satisfy almost all requirements set by Lab Entry's SPC charts. In some cases chart libraries required some additional scripting or installing community developed plug-ins to achieve certain functionality. In addition to satisfying requirements, library's popularity among developers was also taken into consideration. A directional measure for determining popularity was conducted by counting projects in GitHub and checking how many questions

related to a library have been asked in developer community site Stack Overflow. It should be mentioned, however, that this type of measurement favors greatly open source libraries over commercial libraries, which typically have their own support channels. This is very noticeable in case of Canvas.js, whose popularity numbers are somewhat uncomparable to the rest of the libraries. Performance numbers had a lot of variation among libraries, also depending greatly on the chart type, enabled features, and its complexity. However, its quite safe to say that performance won't became a problem in any case as the lowest count for performance issues was found to be 16 000 data points with dc.js chart.

Finally, after weighting benefits and drawbacks for many libraries, the selection for Lab Entry's SPC charts was chosen to be D3.js. To its benefit, D3.js satisfies all requirements and has large community support behind it. The main design ideology with D3.js is to provide low-level tools for implementing different kind of chart components. This also means that D3.js doesn't actually have any pre-built charts in itself, but relies on community's contribution for implementing and sharing them. Although using low-level API for building charts from the very beginning has quite steep learning curve, it was estimated that the initial extra work invested in the development would pay off in the future when new features are added. [45]

5.1.4 Interface rendering

Most of the modern web pages are dynamic in nature, meaning that their content is determined at the moment of loading, for example based on user inputs and database data. This is the opposite of static web pages, where content stays the same regardless of who makes the request and when. Dynamic web pages can be divided to two categories depending on whether the rendering is done on the client or the server side. Examples of server side scripting languages are for example PHP, ASP.NET and Perl. On the client-side, basically all rendering is based on JavaScript, although multiple different libraries are available to choose from, for example Mustache, Handlebars, EJS, Underscore and Lodash. [46]

Lab Entry's interface rendering was decided to be done on the client-side with a library called Lodash [29]. The benefits for using Lodash is that it supports both AMD and NPM modules, which means that it can be used both client and server side. Although all rendering is done on the client side as page is loaded, Lab Entry's calculator feature makes use of Lodash's templating engine as well, parsing user

formulas to runnable JavaScript code, which is then evaluated to calculator result value. As calculator code is basically JavaScript, it can be executed either on browser or on the server in a safe container. A more detailed overview of Lab Entry's calculator feature can be found in section 5.2.2.

5.1.5 Bootstrap and jQuery

One of the most important front end framework selections for Lab Entry was Bootstrap, which provides tools for building dynamic web pages for different size displays. Main reason for using Bootstrap came from the idea that Lab Entry should be useable also with a tablet, meaning that interface elements need to arrange differently when resolution and screen size are restricted. Other useful features used from Bootstrap were dropdown menus and modal element. [47]

Bootstrap utilizes another very commonly used JavaScript library jQuery, which provides responsiveness and scalability for bootstrap elements. Additionally, many prebuilt components have been developed over jQuery, where few used were *Bootstrap Datepicker* [48] for selecting analysis timestamp values and previously mentioned datatables.net. In itself, jQuery was also used in many occasions, where user interface (DOM elements) needed to be accessed from within the components' AMD modules. [34]

5.2 Back end

Back end has been entirely built on Node.js, which is a server framework for JavaScript. Technology selections were mainly focused on well-tested and popular libraries and frameworks, which have proven their capabilities with similar projects within Valmet and elsewhere. In applicable cases, libraries which could be used on both client and server side were favored over those which could be used only on the server environment. This section takes a review of the most important technology selections which have had profounding influence on the application.

5.2.1 Node.js with Express.js

Node.js is a HTTP server framework based on JavaScript and built on V8 JavaScript Engine. V8 is Google's open source project and is also used as Google Chrome's

JavaScript runtime. Currently supported operating systems are Windows 7 or later, various Linux distros and macOS 10.5+. The power of Node.js comes from its event-driven, non-blocking I/O model, which means that all code is executed asynchronously. With this approach it is easy to implement concurrent applications, which is essential for web servers where multiple clients have to be served simultaneously. Some web server frameworks use threads for concurrency, which generally causes some CPU overhead and requires implementing mechanisms for resource sharing and avoiding process dead-locks. This, however, is not a problem for Node.js as all execution is run on a single event thread, where execution is switched quickly between different client request handlers, leading to perceived concurrency. The downside for non-blocking execution is that developer needs to manage execution order by using callback methods, which are called once operation or function's execution is about to end. [49]

While Node.js in itself is fully equipped for web server development, some frameworks have been developed on top of Node.js for maintaining routing and serving HTTP client requests. These frameworks typically take different approaches towards web server structure and what type of clients are being served in general. For example, *Restify* framework's main purpose is to provide tools for building semantically correct RESTful web services, while minimally addressing browser specific needs [50]. Currently, one of the most popular web frameworks for Node.js is *Express.js*. Express.js contains multiple plugins for parsing client requests, managing authentication and server-side client rendering for example. Due to its flexibility and well established position among other web frameworks, Express.js has been adopted to Lab Entry application for increasing development productivity. [51]

5.2.2 Calculation environment and VM2

One of the Lab Entry's most ambitious features is to provide flexible and versatile calculation environment, where user may write arbitrary formulas, containing process measurements, manual input values, physical constants and lookup tables. In order to achieve such features, a sophisticated interpreter is needed, which is able to understand basic arithmetic operations, calculation precedence and even manage hard coded constants (variables). Fortunately, the problem at hand is greatly simplified with the aid of JavaScript. As JavaScript in itself is interpreted language, it is possible to dynamically parse formulas that have been written with JavaScript

syntax. Following code example provides the basic idea behind JavaScript calculator when using an example calculator definitions provided in Table 5.2.

Table 5.2 *Lab Entry calculator example input definitions.*

Description	Variable name	Unit	Type	Test value
Bottle weight full	bottle_weight_full	g	User	151.4
Bottle weight empty	bottle_weight_empty	g	User	98.4
Liquid volume	volume	ml	User	798

Calculator formula in JavaScript:

```
//Calculate density
({{bottle_weight_full}} - {{bottle_weight_empty}})/{{volume}}*1000
```

This will evaluate after value substitution to:

```
//Calculate density
(151.4 - 98.4)/798*1000
```

and finally give calculation result: 66,41.

With this very basic example it is possible to demonstrate the internal working of the calculator interpreter. The first step in the beginning of evaluation is to parse the calculation code template string. Variables in the calculator formula are identified by double curly brackets: {{*variable_name*}}. Parser engine used here is the same as used in interface rendering, Lodash. The second step is the interpolation of calculation formula, meaning that variables get substituted with their respective values. These may either come from database, from user inputs or from default tests values. Finally, the resulting formula string is given to JavaScript parser, which then will evaluate the statement, giving out the final result for calculation. Due to the fact that both client and server are using JavaScript as their programming languages, described steps can be executed on either side.

Although this approach is relatively simple to implement, it has some pitfalls which have to be addressed before calculator can be used safely. Despite the fact that

calculator feature is using just a subset of all existing features of JavaScript language, it doesn't still mean that they won't be available for users to call if so desired. This poses a potential security concern especially on the server side if user's 'untrusted' code is executed without restrictions. At least two major concerns exist here: compromising server security by accessing resources or crashing the server by writing infinite loop as the calculator function. In order to avoid such scenarios, a library called VM2 has been utilized for running user's calculator codes. A general overview of this behavior is shown in Figure 5.1.

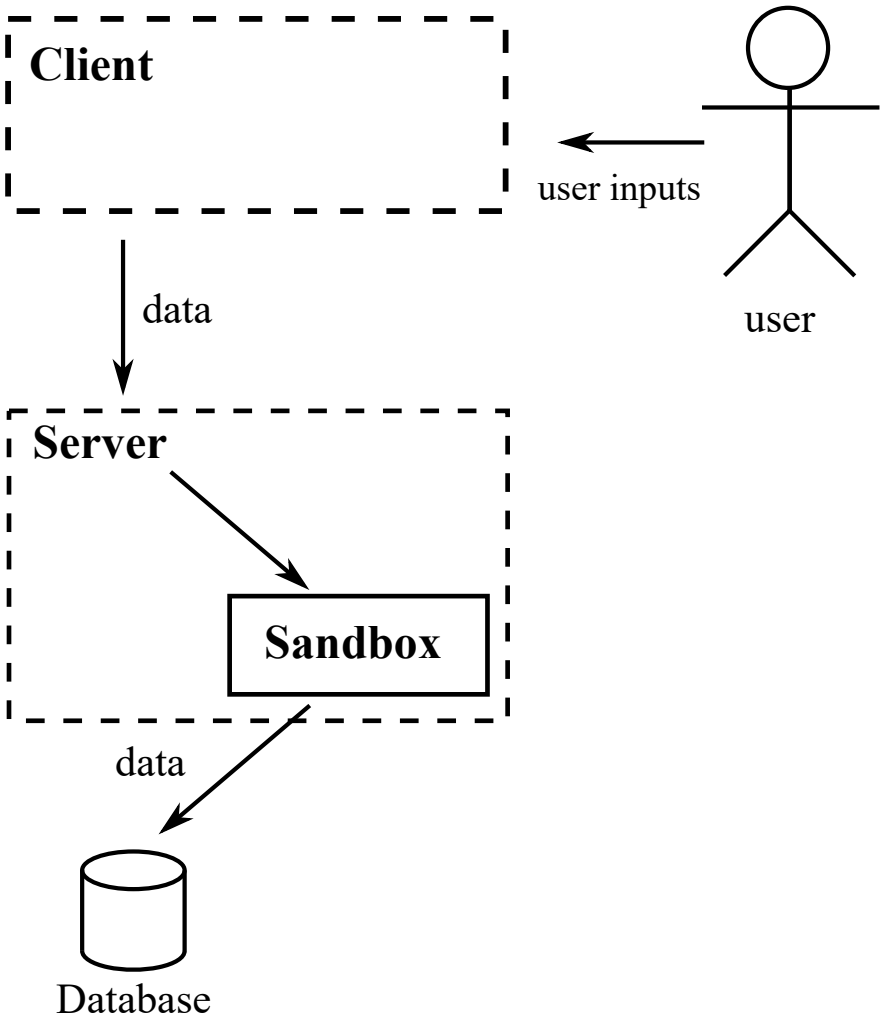


Figure 5.1 *VM2 sandbox executing user function.*

Basically, VM2 provides a sandboxed environment, where executed code has no way of accessing outside application space. This effectively rules out, for example, initializing other processes or accessing file system or database. Another problem,

infinite loop, which causes high CPU load or memory consumption, can be handled by giving code execution a maximum time after which it will get terminated if no valid result has been evaluated. Since calculation functions are quite simple, this maximum time can be kept very short, meaning that the effect on the overall server load remains negligible even if the infinite loop is attempted. [52]

5.2.3 Error tracing and recovery

In order to make it easier for developers to trace bugs and network problems, it is essential to have comprehensive error handling and logging for any irregular behavior. For this purpose a library called Log4js was selected due to its simplicity and verbose logging options. For example, with Log4js it is possible send logs via email, print them to console, write them to a log file or implement custom log writer methods. It is also possible to use additional tags to indicate the type or seriousness of the message that is being logged. Available log tags are *trace, debug, info, warn, error* and *fatal*. Log level can be adjusted with additional option which suppresses all lower level messages. [53]

Since the use of two different databases, calculation executor and other business logic brings additional complexity and potential points of failure on saving analyses, Log4js library is also used for saving analysis input data for later recovery, should any of the mentioned operations fail. This feature brings additional backup option so that none of the analysis data is lost due to failure in any of the application's components. Format for the recovery log file is CSV, which can be opened in Excel for example.

5.3 Databases

As already addressed in Software Requirements chapter 3, Lab Entry utilizes two databases: Microsoft SQL Server and DNA Historian. Since Microsoft SQL database has held its position in top 10 most commonly used databases for long, it has several database driver implementations available for Node.js. After some research, the most suitable library for Lab Entry turned out as a library called Tedious.js. As earlier discussed in Security chapter, Tedious.js implements database query parameter sanitation automatically, effectively blocking SQL injections. Although being simple and straightforward to use, Tedious.js is missing concurrent connection handling,

which is a mandatory requirement for Lab Entry's database driver. Fortunately, this problem was resolved with a wrapper library called *tedious-connection-pool*, which manages database queries by opening multiple database connections and recycling them without the need to open and close connections for every query. Using of connection pooling proved to have a great impact on overall application performance. [37] [54]

Interfacing with second database, DNA Historian, turned out to be a more complicated task due to the fact that the only potentially useable interface available in DNA Historian was ODBC (Open Database Connection) protocol. After spending some effort for searching suitable implementations of Node.js ODBC database libraries, it was agreed to take alternative approach by interfacing with DNAData instead. As discussed before, with DNAData's Web Service interface it became possible to access DNA Historian without the need of ODBC. Another advantage of using DNAData was the possibility to make use of DataClass methods, which can execute more complex calculations, removing some of the calculation load from the Node.js server. In practice, interfacing with DNAData was implemented by developing so called DNAData Interface Server, which acts as a mediator between Node.js and DNAData. A general overview of Lab Entry's back end with DNAData is presented in Figure 5.2.

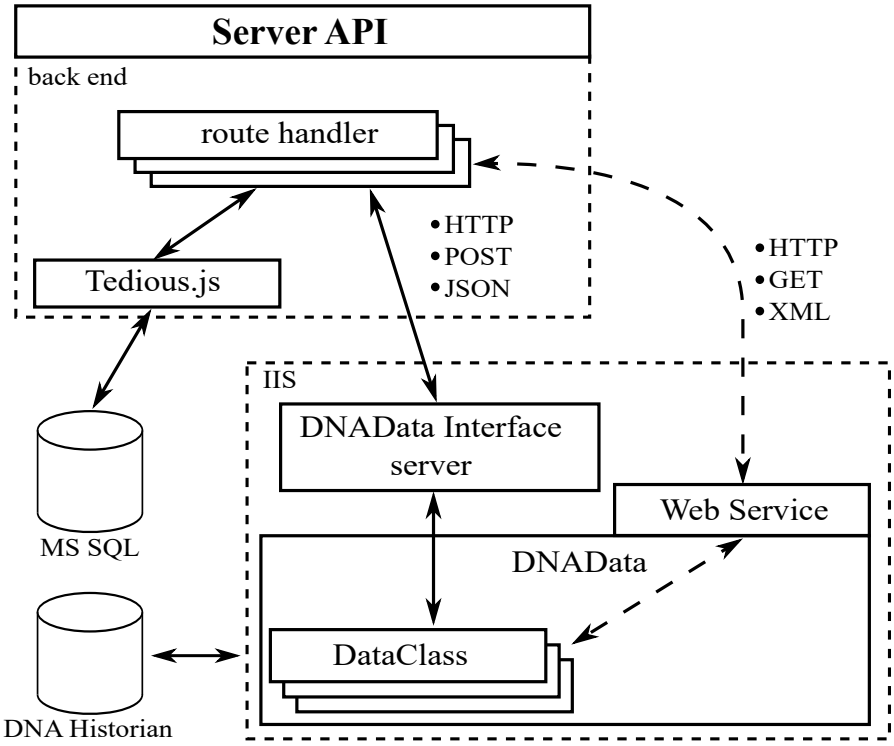


Figure 5.2 *Lab Entry's back end with DNAData interface.*

As Figure 5.2 shows, connection to DNAData and DNA Historian is made through an additional proxy server, which handles incoming DNAData calls and converts them internally to DataClass methods calls. DNAData Interface server provides a JSON interface, which is an alternative to XML-based Web Service interface provided by DNAData. Reason for this seemingly complicated approach of creating a proxy server came from the fact that Node.js supports natively JSON format as its most basic form of data, making sending and receiving HTTP messages a trivial task in the server's perspective. DNAData interface server was built on Microsoft IIS, meaning that it integrates easily on existing DNAData installation, which also makes use of IIS.

6. RESULTS

Throughout the development project of Lab Entry application comprehensive study on SPC theory has been conducted, multiple different technologies have been researched and large amount of internal and customer feedback have been collected and analyzed. This chapter introduces the final product and its most central features, focusing on how previously mentioned aspects have had their effects on it. Features and screenshots in this section are introduced as they have been implemented in the customer pilot version of Lab Entry. Main focus is on the most essential features, which were listed in software requirement specification's Table 3.1. Second topic in this chapter is how application validation was conducted and what results were obtained. These are, for example, how well implemented features met customer expectations and if not, what improvements should be made.

6.1 Lab Entry overview

First contact to Lab Entry application for user is the login page, where all users are identified by their personal or shared user account. Lab Entry's authentication relies on Valmet DNA account management and in practice this means that users may use their existing DNA Report portal credentials when logging in to Lab Entry. In addition to this, assignment of user specific roles is possible, so that system administrator may conveniently manage what actions users may perform in the system. In the pilot version, even though application recognizes user roles, actual features utilizing them are only for testing purposes. Fully integrated role management is planned to be implement in the future. Lab Entry login page is shown in Figure 6.1.

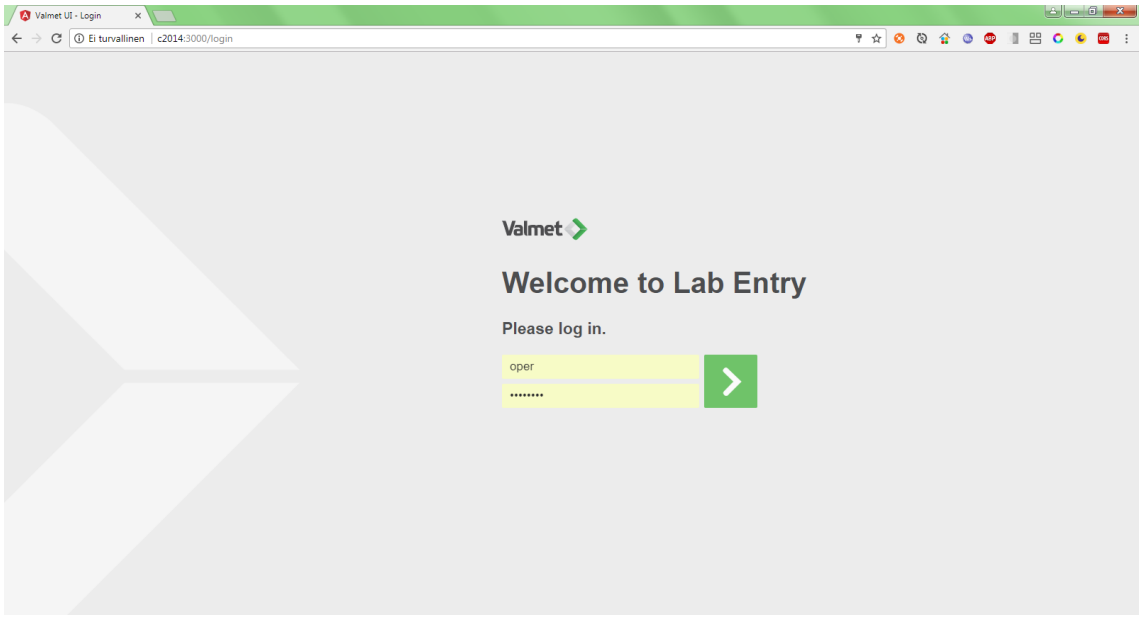


Figure 6.1 *Login page with DNA authentication.*

One of the most central features for laboratory staff is to have the possibility to easily search for specific analysis among hundreds of configured analyses. For this task Lab Entry has adopted two level categorization from DNALab for analyses: work groups on top level and and displays on the second level. The actual hierarchy can be seen in the left side of Figure 6.2. Opening of tree nodes in the hierarchy causes analysis list to be filtered based on work group or display, depending on the depth of opened node. Additionally, text based search can be done, where search term is targeted to almost every attribute displayed in analysis row like analysis name, tag name, latest value and entry date for example.

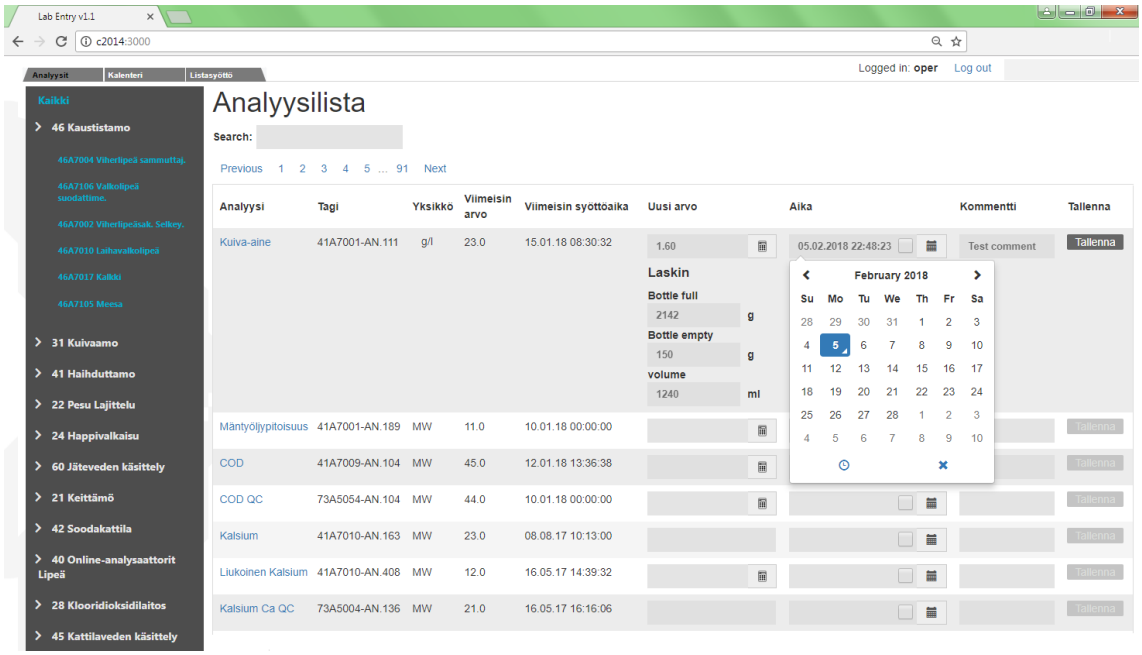


Figure 6.2 *Analysis list with navigation and inline calculator.*

Figure 6.2 also shows the new inline calculator feature, where user can click calculator icon for certain analyses, and open a dropdown calculator with configured input values. Here user may insert values to the input fields, while calculation result gets calculated instantly and inserted to the 'New value' field. Alternatively, if user wishes not to use the calculator, this is also possible by inserting the values directly to the 'New value' input field. Values that have been inserted to the calculator input fields by the user are also saved to the database in case they need to be observed later.

As a new and often requested feature Lab Entry implements integrated SPC charts, which can be displayed in an analysis details popup window by clicking analysis name from the Analysis list view or List Entry views. Additionally, analysis entry history is shown below charts, including any comments associated with the entries. Other displayed information are aligned reference or process measurement value, subtraction between these, and SPC test column where additional information is displayed about automated SPC test results. As explained in theory section 2.7, automated tests are configured based on the theory developed by Lloyd S. Nelson. Using of each automated test is completely optional and configurable, as the relevance of tests may vary between processes. The SPC tab of the analysis details modal is displayed in Figure 6.3.

Figure 6.3 *SPC Charts with entry history.*

The inline calculator feature shown in Figure 6.2 can be configured in the second tab of analysis details popup window. Calculator feature, which is also found in DNALab, has been improved since, as with Lab Entry it is now possible to write templated JavaScript code where calculator source values get substituted in the formula and final results are evaluated in the JavaScript runtime environment. Since calculator syntax is plain JavaScript, it is natively supported by all web browsers as well as Node.js back end. As allowing users to write arbitrary JavaScript code contains a high risk of Injection attack (detailed explanation in theory section 5.2), user inserted 'unsafe' code is executed in server-side virtual container where it

may not access external resources or cause infinite loops causing server to crash. Calculator feature with its configuration view is shown in Figure 6.4.

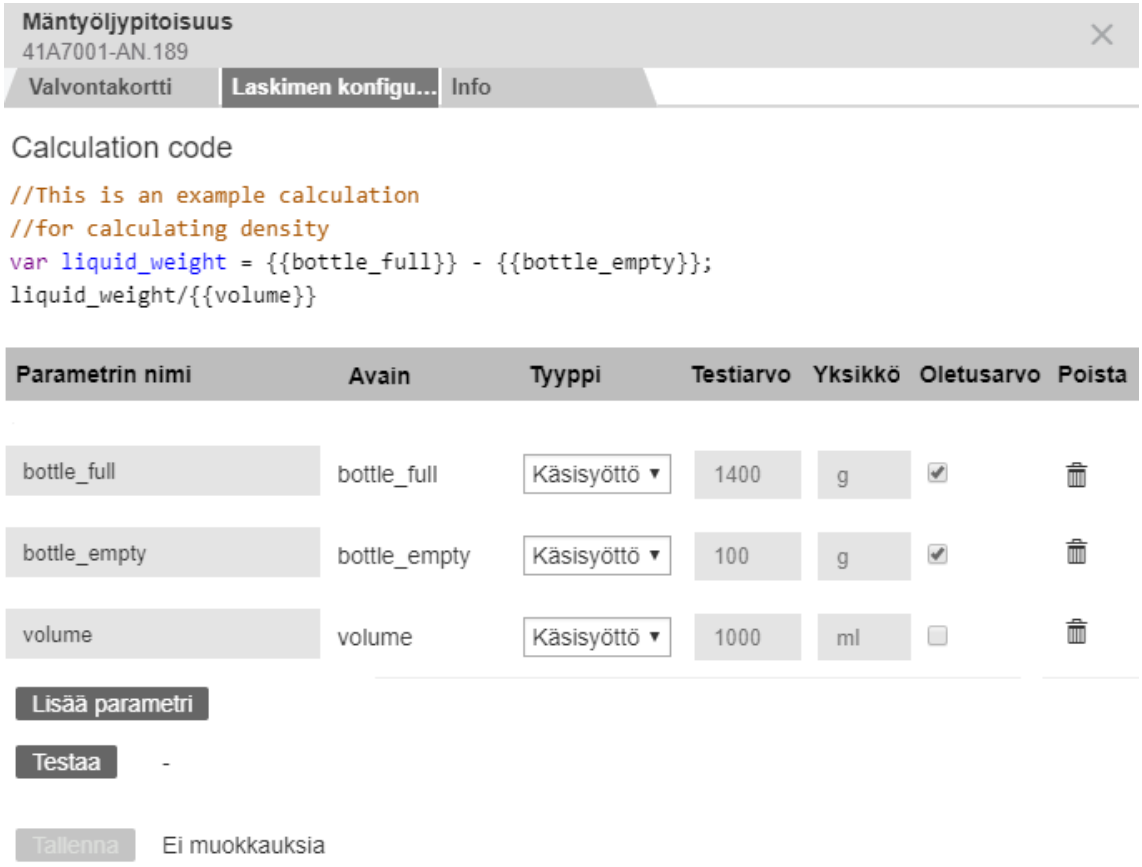


Figure 6.4 *Calculator configuration view.*

In addition to plain analysis value insertion and history tracking, Lab Entry has work planning and assisting features, where user may create recurring calendars for certain analyses. Analyses may be assigned to certain weekdays, but other conditions can be applied also. For example, repeating an analysis on a certain day of a month or once a year on some specific date is also possible. Once user has defined one or more calendars, Lab Entry constructs week programs based on them. Week programs serve as basis for controlling daily routines in laboratory work, while providing flexibility to change planned days if an analysis can't be made for some reason. Daily work flow can be monitored during the day, as completed analyses are updated in real time in the week programs. Week program feature is shown in Figure 6.5 and List Entry insertion page is shown in Figure 6.6.

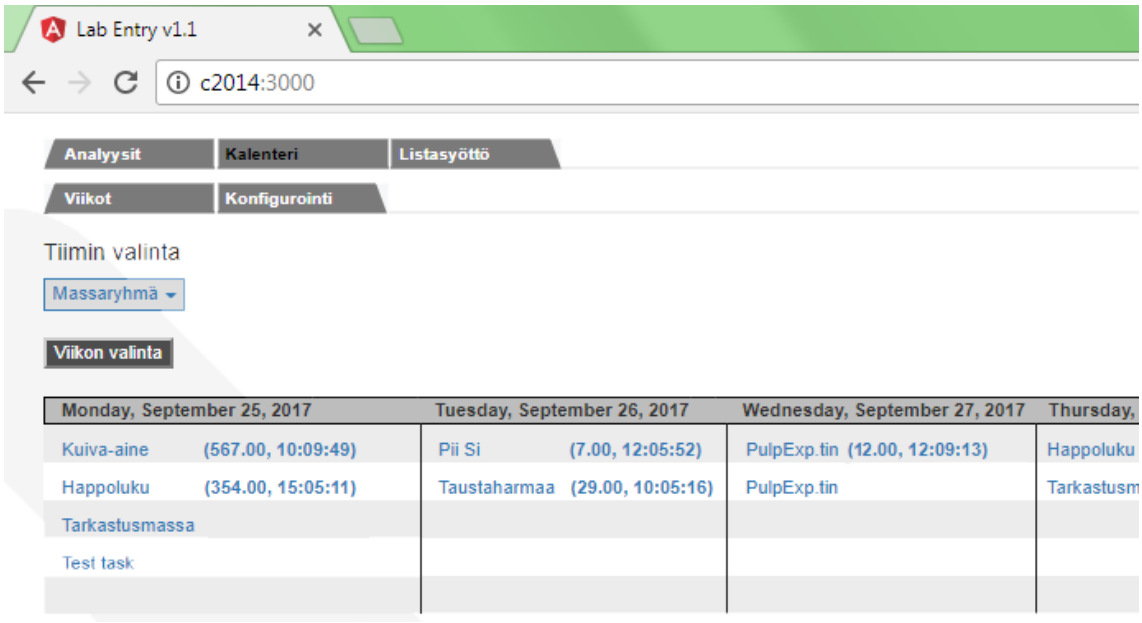


Figure 6.5 *Calendar view with several analyses.*

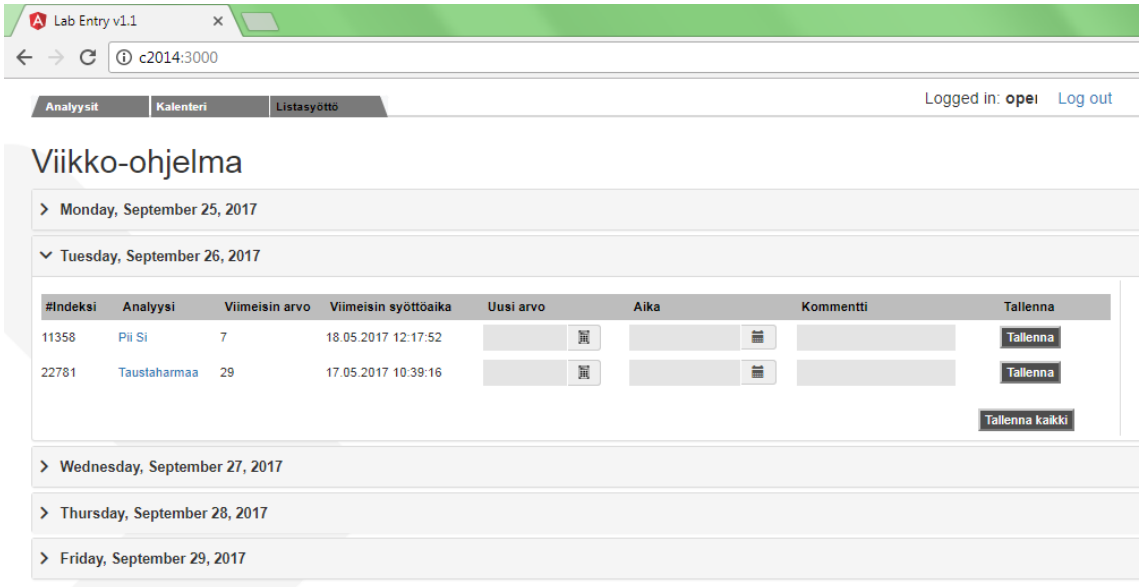


Figure 6.6 *Week Program's List Entry*

The List Entry view shown in Figure 6.6 shows analyses that have been configured for selected week program. The main idea behind this view is to show conveniently only relevant analyses without the need to search for specific analysis among all configured analyses in the application. Inline calculator feature and opening of

analysis details popup features work similar to Analysis List view shown in figure 6.2.

6.2 Application validation

Application validation, which by definition means checking that the application meets the operational needs of the customer, was planned to be conducted in three phases: customer demonstrations, internal pilot and finally customer pilot. Each phase had different goals, as customer demonstrations provided general overview whether application development was heading to the right direction or if it was missing something essential. During internal pilot phase, Valmet Automation's project and application development teams had free access to the development version of the application for finding out bugs and validating that each of the implemented features were behaving as specified. The planned goal for the final customer pilot phase was to develop production ready version of Lab Entry by deploying application to customer laboratory environment and getting first hand experiences from laboratory staff directly. Validation data was collected and reviewed from each phase in order to guide the development process.

6.2.1 Customer demonstrations

Customer demonstrations were conducted in the middle of the development process so that any small or moderate changes in application design would still be possible to implement should any emerge. In total, two different customers participated in demonstrations: pulp mill 1, which also participated in forming initial requirements specification, and a fabrics factory. With pulp mill 1, the idea was to check how well initial requirements, laboratory work and the role of statistical process control, were understood and how well they were implemented in the application. On the contrary to pulp mill, fabrics factory provided completely different perspective to laboratory work with more batch or unit production -like approach. Summaries for both visits are in appendixes 3 and 4.

The most valuable feedback received from visit in pulp mill 1 was that Lab Entry's user interface should have more details and for example, contain information about sample collection place, units, limits and whether analysis can't be made due to an unfavorable process state. Also, the Week Program feature (Figure 6.5) should

have the possibility to add other tasks, like ordering chemicals, in addition to just analyses. In general, Lab Entry received positive feedback in useability and that initial requirements were mainly well understood and implemented.

Even though batch production was originally ruled to be out of scope for Lab Entry, additional visit to fabrics factory was made in order to explore synergies with existing systems and how easily batch production could be implemented in Lab Entry. Also, current development version at the time was demonstrated and notes were taken down of what it would require from Lab Entry to be useable in batch production environment. In general, customer found Lab Entry interesting and especially tablet support was considered to be convenient feature. Most of the improvement ideas were directed towards user interface and its intuitiveness. After the visit, it was commonly agreed that with current implementation it would not be feasible to support batch production at this time. However, later some design choices with Lab Entry's database structures were made in order to better support batch production in the future should this become a feature.

6.2.2 Internal pilot

Internal pilot testing was targeted mainly for Valmet Automation's personnel who had prior experience in laboratory system projects or DNALab development. In practice, testing was conducted by releasing development version of Lab Entry to a test group, where they could freely test any features. Feedback received from the test group proved to be valuable as many of the addressed issues or improvement ideas had occurred before with customer projects or DNALab. Internal pilot's summary is found in appendix 5.

The main topics in internal pilot's feedback were focused mainly on displaying and searching of useful information in user interface. For example, units, limits and sample collection places were all pointed out to be relevant information to be displayed along other information. Also, some additional feature recommendations were made to improve user productivity, like adding button for saving multiple analysis values simultaneously or improving multi-time selection to automatically handle all analyses in current display.

R&D UX (user experience) team was also involved in the development process in order to improve the general useability and also match Valmet style guidelines which

concern all applications. Several layout changes were made to the application, for example grouping analyses in a separate week program view as shown in Figure 6.5 and implementing the calculator feature as an inline view instead of separate modal.

6.2.3 Customer pilot

The real test for the Lab Entry application in an actual laboratory environment will be the upcoming customer pilot. Unfortunately, for the time being, the state of the application's development was found to be still too incomplete in order to have a stable and safe deployment to customer's automation environment. Several tasks related to verification, overall reliability, and polishing the end user experience were still waiting to be completed.

Expectations towards the results in the customer pilot are the validation of basic functionality, covering the inserting, editing, deleting, and commenting of analysis values, as well as the SPC functionality with single selected SPC chart. For this purpose a chart type of I-MR was chosen based on the wishes from pilot customer and due to the fact that its operational correctness can be validated against currently used SPC product called *X-Chart*. Secondary objective is to let the users to test the development versions of Calendar and Week Program features, as these won't need to be fully implemented in order to be useful for the users.

6.3 Future development plans

Based on previously mentioned feedback and other findings during the project some tasks and development plans have been drafted. This section contains the most central development tasks which will be completed in the near future.

6.3.1 Lab Conf

As it has been mentioned in several occasions, Lab Entry with its current implementation utilizes DNALab's configuration application to manage hierarchy, analyses and related attributes. Such approach was taken in order to manage the workload of implementing the application. However, this temporary solution has its own downsides in the form of utilizing and supporting two separate applications, thus needing a better and more permanent alternative soon.

In contrary to DNALab, it has been planned that configuration features will get integrated directly to Lab Entry, without the need to have two separate applications. Some of the configuration tasks have already been implemented, as the new features, like Calendar, Week Program, and the new Calculator, have already required the development of their own configuration views. Once Lab Entry has achieved complete independence from DNALab, it also becomes possible to have better influence on the database structure without the risk of breaking configurations.

6.3.2 Application template

Before starting the development of Lab Entry application it was already known that other applications like DNALab with similar needs for updating exist at Valmet Automation. Thus, one goal in the development of Lab Entry was to provide a proof of concept for implementing applications with selected web development techniques. As a very important side product of Lab Entry development, an application template was developed where only the most essential parts were included in a form of a simple demo application.

The main features in the template application were demonstrating the component concept (module pattern) used in Lab Entry, data queries and writes from DNA Historian and MS SQL servers and the publish–subscribe pattern used for updating all the clients with latest server data. These features were selected based on the fact that they cover most of the use cases in Lab Entry and similar applications. In the future, it is possible to further continue the development of this template application based on updates to Lab Entry or other applications with similar structure.

6.3.3 SPC features

As The Figure 6.3 displays, Lab Entry's pilot version has an integrated SPC chart with Average and Range charts. However, as the review in the theory section shows, there are more options for configuring different type of charts based on various statistical methods, bringing the user a better possibility to monitor the process. These options should be further studied and integrated as a part of the application.

Additionally, the automated monitoring tests, which were introduced in theory section 2.7, have not been yet fully implemented. Instead of hard coding the tests

directly to the application, plans are to create a more generic interface, where users may write custom test methods for detecting patterns in the data. This feature would work similar to the calculator feature, which was introduced as a new feature in Lab Entry.

6.4 Project evaluation

The application development project has been a learning process in itself. Many new project management tools and techniques, as well as development methods and environments were tested during the project – some more successfully than others. This section evaluates which methods were successful, which didn't work and which could be improved. Other subjects for evaluation are the project scope, collaboration with team and the customer, and technology selections.

First observation about the project timeline was that at some point of development it was found to be overly optimistic. The root causes behind the delays in development work can be traced back to extra time needed to learn some of the new development techniques, like Node.js, and to other projects, which lead to temporary reductions in development resources. Third reason for extra work done with development was due to the insufficient understanding of the complexity of the whole application during design phase. This lead later to the need for refactoring of the application back end to a more maintainable structure. Additionally, web development techniques are changing quickly and some older design patterns, like callbacks for managing asynchronous execution, were replaced with newer Promise pattern during the development. On positive side of noticing evident delays in development, project scope and timeline were readjusted reasonably, so that development would continue without significant delays.

Technology selections have proven to be successful in general, but there are some technologies which could have been beneficial for the development if adopted early on to the project. These, however, had to be discarded due to having too high re-factoring costs. For example, the Node.js back end would have benefited from using TypeScript instead of plain JavaScript. The fact that TypeScript contains classes, and uses compile-time checks for object attributes, would have saved a lot of trouble for checking runtime errors for missing object properties. Second unnecessarily complex design choice was to access database tables through stored procedures, which would hide the details of SQL scripts behind simple and well

defined interfaces. The initial idea behind this choice was to make it possible to later switch to completely another database, for example PostgreSQL, and still keep the impact to Node.js back end minimal. In practice, however, frequent SQL script updates during development would have been managed more efficiently by using hard coded SQL scripts in Node.js back end instead of running procedure update scripts to SQL server.

Collaboration with Valmet R&D was found to be highly beneficial for the Lab Entry development as some structural and technological selections were done based on the meetings and code audits done with R&D's developers and UX team members. Additionally, R&D gave guidance on how DNAData interfacing should be implemented without compromising stability, security or performance of either systems. Other ideas adopted from the R&D were Agile development methods and continuous integration tools for running automated tests and building demo versions of the application.

7. CONCLUSIONS

The purpose of this thesis work was to design, implement and validate a new laboratory analysis entry system based on the knowledge gained from researching statistical process control, interviewing customers and Valmet employees, and exploring different implementation techniques. Research methods used during the project can be divided to literature, the Internet and interviews. Literature sources served the main purpose when researching statistical process control, Internet sources were used for exploring different development techniques and interviews provided essential information during application specification, as well as validation phases.

The thesis work results consist of application overview and validation, and evaluation of the development project. Additionally, the most important development tasks for the future were planned ahead to be implemented after the thesis work project has concluded. In general, the implemented application was found to fulfill the functional requirements defined during the project. These were, for example, inserting, modifying and deleting analysis entries, analysis commenting, using inline calculator, planning daily work with calendar and monitoring the process state with statistical process control charts. Secondary objective, developing a demo application serving as a template for similar development projects, was also completed. Challenges encountered during the development were mostly related to keeping planned timetables and minor code refactoring tasks on the application code base to ensure maintainability for the years to come.

APPENDIX 1: VISIT AT PULP MILL 1

Visit goals

- Research laboratory work in pulp industry
- Receive feedback and improvement requests about current laboratory system

Current system

- Entry system: DNALab, installed 2014
- Work divided systematically to day, week and monthly 'baskets'
  - Idea is to pick analysis, complete it and log results
- Work is done flexibly and in a self-organizing manner

Daily workflow

1. Construction of daily tasks based on week program
2. Start work by picking (any) analysis from day basket
3. In work station write down to notebook any source data (bottle weight, volume, etc.)
4. Start analysis, this may take some time depending on equipment
   - It's possible to do multiple tasks concurrently.
5. After completion write results to notebook
6. Write results to analysis related excel containing full history for analysis
7. Write results to DNALab
   - Use DNALab's calculator as needed.
8. Check analysis' SPC charts for failed test results

9. Write analysis result to week program excel to indicate task has been completed.
    - Optionally comment analysis or indicate if analysis could not be completed for some reason.
10. Send email about analysis to people who are interested about it
    - Separate mailing lists for all entries and failed tests

Important features

- DNALab's calculator
- SPC tests
- Analysis hierarchy by factory sections for finding correct entry display

Improvements and wishes

- Sometimes analysis calculation's source data is dependent on another calculation's result. How this could be handled when inserting multiple analyses simultaneously?
- Inability to comment analyses if no valid value could be done.
- Analyses have limits. Currently they aren't displayed in entry display.
- Week programs should be integrated in (DNALab) Entry and it should provide better guidance in terms of workflow than plain value insertion.
- Week program should contain basic calendar functionality for managing events.
- If due to process state or some other reasons it's not practical to make an analysis, this could be indicated somehow.
- It would also be nice to have notification when it is possible to take certain rare samples based on process state (mass being pumped from tower once per month).
- In case of multiple concurrent lines, analysis should be automatically inserted to the position corresponding active line at the moment of taking sample.

- Entry calculator feature should be extended to cover more complex functions and value look-up-tables (NaOH concentration based on temp and density, which in return is based on volume and mass)
- Week program should contain link to analysis manual. This is convenient for new employees.
  - Input fields for source values
  - Predefined hard-coded selections (certain bottle with known volume).
  - Calculation source data should be saved for later inspection in case of deviations.
- SPC charts should be integrated to entry display & week program to conveniently view test results.
- Automatic signaling (email) as new analyses are inserted.
  - Configurable lists
  - For every new analysis
  - For certain failed tests

APPENDIX 2: VISIT AT PULP MILL 2

Visit goals

- Research laboratory work in pulp industry
- Receive feedback and improvement requests about current laboratory system

Current system

- Entry system: DNALab, installed 2008
- 5 + 1 employees, normal day work
- Work circulation 2 weeks
- Day programs for assigning tasks

Daily workflow

1. Pick analysis, also verify that it can be made.
2. Complete analysis and write down results to workbook.
3. Calculate analysis results from source information. This is done with plain pocket calculator.
4. Analysis insertion and (commenting if needed) to Excel sheet.
5. Excel is emailed to interested persons. List is looked-up from a folder.
    - New values
    - If value is off-limits
    - Call to shift manager if value is off-limits
6. Analysis insertion to DNALab. Some process measurement alignments don't work. They have to be looked-up manually from trend.

Observations

- No SPC tests or similar feature
- No utilization of DNALab's calculator feature
- Laboratory staff manages also chemical orders.
- Calibration need is reasoned from limits only → call to instrument installer if needed.
- Exceptions to daily routines if disruptions occur in the process → need for additional analyses.
- Samples are retrieved personally, by process staff or by pneumatic tubes.

APPENDIX 3: 2ND VISIT AT PULP MILL 1

Visit goals

- Demonstrate Lab Entry development version and receive feedback
- Get more improvement ideas about laboratory work

Demonstration

Improvement ideas

- User should be able to see analysis position, sample collection place and unit.
- In week program, specific analyses should be labeled based on current status. For example, taking analyses is not reasonable if process is not active.
- Assigning tasks to specific persons or possibility to comment analyses in general.
- Dividing week programs in different teams and possibility to filter view based on these.
- Week program should have option to add generic tasks which are not analyses to support laboratory work planning in general.
- Clear indication if limits have been exceeded
- Calculator formula is not important to show once it has been configured and taken to use.
- Calculation order should be planned carefully as some analyses are dependent on each other.
- Keyboard support: navigation with arrow keys and saving analysis with double enter.
- Multi-time selection for many analyses with same timestamps.

Positive feedback

- Integrating SPC tests closely to analysis entry page was considered as an excellent feature.
- Tablet support was found to be a good idea.
- Week program with configurable calendar templates was useful feature.
- Deleting of accidentally inserted analyses from SCP page and directly observing results was nice addition.

APPENDIX 4: VISIT AT FABRICS FACTORY

Visit goals

- Research laboratory work in batch production environment
- Demonstrate Lab Entry development version and receive feedback

Current system

- SAP, installed early 2000's

Daily workflow

- Laboratory workflow is mainly driven by external factors
  - Customer orders
  - Quality control
  - Internal work orders from R&D
- Some tasks are assigned automatically based on pre-defined rules in SAP system.
- Sample count and tolerances are calculated from total batch size.
- Every production phase has a set of parameters which have to fall within tolerances.
- Exceeded tolerances may result in discarding of the product. Information is passed via email.
- Every measurement, comment and attachment is saved in SAP.

Observations

- SAP has an essential role throughout whole manufacturing process and it has been continuously improved during its 20 year time span.
- Every task has a link to unique customer order number.
- Automation process history is not collected in SAP and no process measurement alignments are made.
- Analysis collection is not bound to timescale but in product or batch.
- Analyses have rich commenting and attachments, typically customer reports or images from electron microscope.

Demonstration

- The importance of intuitive user interface was emphasized
- Common tasks, like inserting new analyses, should be as easy and fast as possible.
- Problems and exceeding of tolerances should be indicated clearly.

APPENDIX 5: INTERNAL PILOT RESULTS

Features

General notions

- It would be nice to be able to search from all analyses comments.
- Validity checks for inserted values: it should not be possible to insert values exceeding preconfigured absolute limits.
- Comma as a decimal separator didn't work. This should be also possible.
- It was found that deleting inserted values was not possible.
- Scroll bars were found to be too narrow to be used conveniently.

Analysis list

- Unit column is missing
- Column name *'Tag'* should be named *'Position'* or *'Variable'*
- Workgroup (työryhmä) should be workset (työkokonaisuus)
- Display column should be visible.
- When clicking tree hierarchy nodes, analyses were filtered based on node keyword. However, this was problematic when keyword appeared elsewhere: for example, analysis name '*KappaBright*' was matched when opening display with name '*Kappa*'.
- How the need for calibration or limit exceeds are indicated in the user interface?
- Time multiselection should be easier, maybe selecting times for all displayed analyses with single datetimepicker?
- Button for saving all inserted analysis values simultaneously was missing.
- Analysis comments should have maximum value (which is configured to database).

Calendar

- Week picker component should be improved. Maybe buttons traversing weeks one-by-one?
- Plain analysis name (f.e. '*Kappa*') is not descriptive enough as multiple analysis may have same names. Maybe adding sample place could bring distinction between analyses?

List Entry

- When opening List Entry, current day should be opened automatically.
- Daily section didn't open when arrow '>' button was clicked.
- Datetimepicker opened partially outside of page viewport when close to bottom.

BIBLIOGRAPHY

[1] *Pulp and Paper Industry*, 2016, website (visited on 30. 08. 2016), URL: https://www.forestindustries.fi/statistics/pulp-and-paper-industry/.

[2] *Metsä Group to build next-generation bioproduct mill in Äänekoski*, Apr. 2015, website (visited on 09. 09. 2016), URL: http://biconsortium.eu/news/metsa-group-build-next-generation-bioproduct-mill-aanekoski.

[3] *Biotuotetehdas - Metsä Group*, 2017, website (visited on 28. 10. 2017), URL: http://biotuotetehdas.fi.

[4] J. S. Oakland, in: *Statistical Process Control*, 6th Edition, 472 pages, Oxford: Butterworth – Heinemann, 2008, ISBN: 978-0-7506-6962-7.

[5] D. S. Chambers and D. J. Wheeler, *Understanding Statistical Process Control*, 2nd Edition, 406 pages, SPC Press, 1992, ISBN: ISBN 0-945320-13-2.

[6] C. Berardinelli, *A Guide to Control Charts*, website (visited on 26. 05. 2017), URL: https://www.isixsigma.com/tools-templates/control-charts/a-guide-to-control-charts/.

[7] D. J. Wheeler, *Advanced Topics in Statistical Process Control*, 1st Edition, 370 pages, SPC Press, 1995, ISBN: 0-945320-45-0.

[8] *Nelson Rules*, website (visited on 09. 11. 2017), URL: http://leansixsigmadefinition.com/glossary/nelson-rules/.

[9] Ambysoft Inc, *Agile Requirements Modeling*, 2012, website (visited on 18. 05. 2016), URL: http://agilemodeling.com/essays/agileRequirements.htm.

[10] Valmet Oy, *Valmet DNA*, May 2016, website (visited on 10. 05. 2016), URL: http://www.valmet.com/products/automation/valmet-dna-dcs/.

[11] Valmet Oy, *Valmet DNA CR DNAdata Developer's manual*, unpublished manual, 2016.

[12] Valmet Oy, *Valmet DNA historian*, website (visited on 27. 05. 2017), URL: http://www.valmet.com/products/automation/valmet-dna-dcs/valmet-dna-products/operator-tools/valmet-dna-historian/.

[13] Microsoft, *XML Web Services Basics*, Dec. 2001, website (visited on 10. 05. 2016), URL: https://msdn.microsoft.com/en-us/library/ms996507.aspx.

[14] P. Kotiluoto, *DNAData Development Overview*, Valmet Automation, unpublished manual, Nov. 2010.

[15] *metsoDNA IA Laboratory Data Management DNAlab*, unpublished manual, Metso Automation, May 2001.

[16] M. Rintala, *DNAlab kehitystarpeita ja ehdotuksia*, unpubished manual, Dec. 2015.

[17] A. Stellman and J. Greene, *Applied Software Project Management*, ed. by A. O. Mary T. O'Brien, O'Reilly Media, Inc., 2006, ISBN: 978-0-596-00948-9.

[18] *Improve Your Understanding of Web Applications*, Aug. 2017, website (visited on 12. 12. 2017), URL: https://www.lifewire.com/what-is-a-web-application-3486637.

[19] *Usage of server-side programming languages for websites*, Sept. 2017, website (visited on 09. 09. 2017), URL: https://w3techs.com/technologies/overview/programming_language/all.

[20] A. Aggarwal, *REST API – Back to Basics*, May 2017, website (visited on 30. 01. 2018), URL: https://restful.io/rest-api-back-to-basics-c64f282d972.

[21] M. Anicas, *5 Common Server Setups For Your Web Application*, May 2014, website (visited on 28. 01. 2018), URL: https://www.digitalocean.com/community/tutorials/5-common-server-setups-for-your-web-application.

[22] H. Eronen, *IaaS, PaaS, SaaS? Mikä pilvipalvelu sopii yrityksellesi*, Mar. 2016, website (visited on 01. 02. 2018), URL: https://blog.planeetta.net/iaas-paas-saas.

[23] *What is a web server?*, Dec. 2017, website (visited on 20. 12. 2017), URL: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server.

[24] *What is a URL?*, Dec. 2017, website (visited on 20. 12. 2017), URL: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL.

[25] *Introduction to Web development*, Feb. 2016, website (visited on 20. 12. 2017), URL: https://developer.mozilla.org/en-US/docs/Web/Guide/Introduction_to_Web_development.

[26] A. Osmani, *Learning JavaScript Design Patterns*, O'Reilly, 2017, URL: https://addyosmani.com/resources/essentialjsdesignpatterns/book/.

[27] misaal9, *MVC architecture*, Aug. 2017, website (visited on 07. 09. 2017), URL: https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture.

[28] *Backbone - Model*, Sept. 7, 2017, website (visited on 29. 01. 2018), URL: http://backbonejs.org/#Model.

[29] *Lodash Documentation*, website (visited on 07. 09. 2017), URL: https://lodash.com/docs/4.17.4#template.

[30] *RequireJS*, Jan. 2015, website (visited on 19. 10. 2016), URL: http://requirejs.org/.

[31] *Angular Dependency Injection*, Feb. 2018, website (visited on 03. 02. 2018), URL: https://angular.io/guide/dependency-injection.

[32] *Angular*, website (visited on 15. 03. 2018), URL: https://angular.io/.

[33] Siddharth, *15 Most Important Considerations when Choosing a Web Development Framework*, Dec. 2009, website (visited on 12. 12. 2017), URL: http://code.tutsplus.com/tutorials/15-most-important-considerations-when-choosing-a-web-development-framework--net-8035.

[34] *jQuery*, Feb. 2016, website (visited on 04. 07. 2016), URL: https://jquery.com/.

[35] *OWASP Top Ten Project*, July 2016, website (visited on 30. 08. 2016), URL: https://www.owasp.org/index.php/Top10.

[36] *SQL Injection*, website (visited on 12. 12. 2017), URL: http://www.w3schools.com/sql/sql_injection.asp.

[37] M. D. Pilsbury, *Tedious - Overview*, 2014, website (visited on 30. 07. 2017), URL: http://tediousjs.github.io/tedious/.

[38] *Cross-site Scripting (XSS)*, June 2016, website (visited on 05. 07. 2016), URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

[39] *XSS (Cross Site Scripting) Prevention Cheat Sheet*, Feb. 2017, website (visited on 31. 07. 2017), URL: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet.

[40] *Top 10 2013-A2-Broken Authentication and Session Management*, Jan. 2013, website (visited on 10. 09. 2017), URL: https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management.

[41] *Passport features*, Oct. 2017, website (visited on 25. 10. 2017), URL: http://www.passportjs.org/.

[42] *JSON Web Token Introduction - jwt.io*, Oct. 2017, website (visited on 25. 10. 2017), URL: https://jwt.io/introduction/.

[43] *AMD-JS API Wiki*, Dec. 2017, website (visited on 11. 12. 2017), URL: https://github.com/amdjs/amdjs-api/wiki/AMD.

[44] *DataTables Table plug-in for jQuery*, Oct. 2016, website (visited on 19. 10. 2016), URL: https://datatables.net/.

[45] *D3.js - Data-Driven Documents*, Jan. 2015, website (visited on 19. 10. 2016), URL: https://d3js.org/.

[46] *What is a Dynamic Web Page?*, Dec. 2017, website (visited on 10. 12. 2017), URL: https://www.doteasy.com/web-hosting-articles/what-is-a-dynamic-web-page.cfm.

[47] *Introduction - Bootstrap*, Jan. 2018, website (visited on 22. 01. 2018), URL: https://getbootstrap.com/docs/4.0/getting-started/introduction/.

[48] Eonasdan, *Bootstrap 3 Datepicker v4 Docs*, June 2017, website (visited on 25. 02. 2018), URL: https://eonasdan.github.io/bootstrap-datetimepicker/.

[49] *About Node.js*, 2016, website (visited on 19. 05. 2016), URL: https://nodejs.org/en/about/.

[50] *Restify*, Dec. 2017, website (visited on 13. 12. 2017), URL: http://restify.com/.

[51] *Express - Node.js web application framework*, website (visited on 04. 07. 2016), URL: http://expressjs.com/.

[52] P. Simek, *VM2*, Oct. 2017, website (visited on 16. 12. 2017), URL: https://www.npmjs.com/package/vm2.

[53] *log4js-node*, Dec. 2017, website (visited on 17. 12. 2017), URL: `https://log4js-node.github.io/log4js-node/`.

[54] B. Page, *GitHub - tedious-connection-pool*, Aug. 2107, website (visited on 07. 01. 2018), URL: `https://github.com/tediousjs/tedious-connection-pool`.